

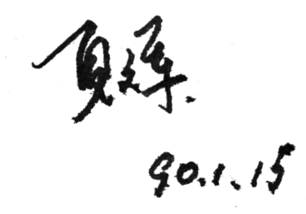
CEC-Ⅰ型中华学习机

用户使用手册



CEC-I中华学习机

用户使用手册



CEC－Ⅰ型中华学习机
用户使用手册
电子工业出版社出版(北京海淀区万寿路)

房山区龙门口印刷厂印装

*

开本：787 × 1092 毫米 1／32 印张： 字数：260 千字
1988 年 9 月第二版 1988 年 9 月第二次印刷
ISBN－7－5035－0237－X／TP29

目　录

第一章
CEC–I 中华学习机简介

CEC– I(China Educational Computer)中华学习机是由我国自己开发的，适合于中小学、家庭及个人学习的计算机。CEC– I 中华学习机的结构灵活，可以根据不同需要扩充功能，如果配上彩色电视机或监视器，再接上软盘驱动器或盒式录音机，就可以构成一台功能较强，具有汉字支持的微机系统。

CEC– I 中华学习机与 APPLE– IIe 微机兼容，其功能与 APPLE IIe 相当，并有所增强，它可以运行 APPLE II 上运行的各种软件，包括数值计算与非数值计算软件，中小学辅助教学软件，以及游戏软件等。主机上有固化的监控程序。BASIC 解释程序，LOGO 子集，汉字管理程序及一，二级汉字库。因此，一开机用户就可以使用这些语言，而不需要从软盘或磁带上读入内存。

CEC– I 中华学习机具有汉字系统，提供拼音和区位输入方式，主机内配有全点阵汉字字库，提供国标一，二级汉字点阵库。

CEC– I 中华学习机主板上具有软盘驱动器，盒式录音机及游戏操纵杆的接口电路，因此可以直接与软盘驱动器，盒式录音机及游戏操纵杆连接使用。

1.1 基本系统的组成

一套完整的微型计算机系统是由几个独立的设备组成的，一般应包括主机，输入设备及输出设备。在中华学习机系统中至少应包括主机与显示器(或电视机)这两种设备，其它设备如盒式录音机，软盘驱动器，打印机，游戏操纵杆等可根据用户的需要进行选配。以下是几种常用的系统配置。

图 1.1 是最基本的系统配置，这种配置由于没有外存储设备，因此只能用于学习或编制一些简单的程序。

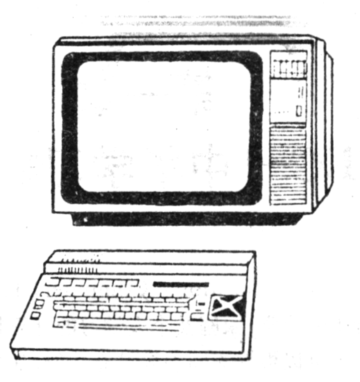


图 1.1　最基本的系统配置

图 1.2 为带盒式录音机的系统配置。在该系统中程序是通过键盘或盒式磁带输入到计算机。程序可以永久地保存在磁带上，因此是一个比较完整的系统配置。其优点是价格比较便宜，但是程序的输入速度较慢，另外程序文件的管理比较麻烦。

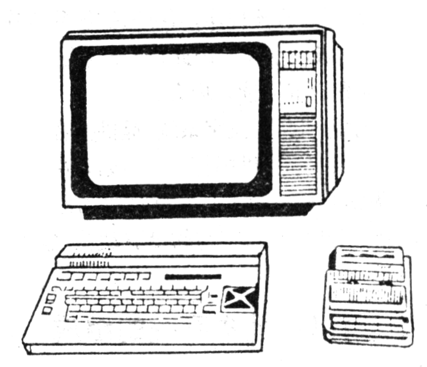


图 1.2　带盒式录音机的系统配置

图 1.3 为带软盘驱动器的系统配置。通过软盘驱动器将程序写入到磁盘上，或从磁盘上读出. 其优点是中华学习机通过磁盘操作系统对磁盘上的程序文件进行自动管理，速度较快，但该系统的价格较贵。

图 1.4 是带打印机和驱动器的系统配置。该系统的配置比较完整，通常被程序设计人员所使用。

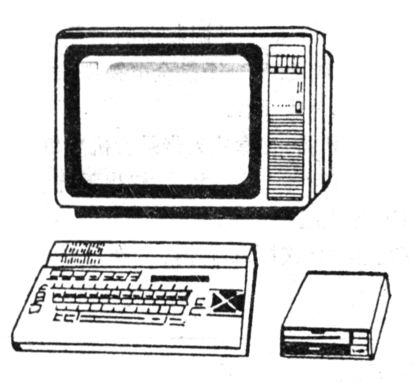


图 1.3　带磁盘驱动器的系统配置

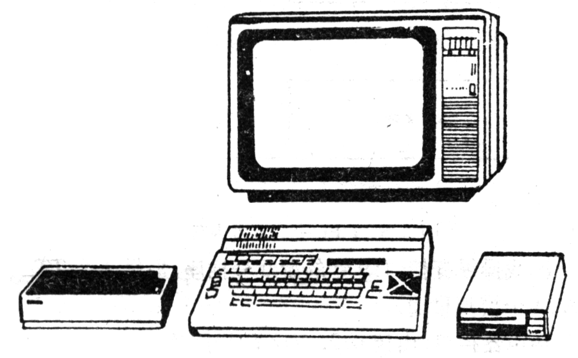


图 1.4　带打印机、磁盘驱动器组成的系统

1.2　CEC–I 的内部结构

CEC– I 主机的内部结构如图 1.5 所示，它主要包括下面一些部件：

① 中央处理器：6502CPU 八位微处理器。

② 随机存储器(RAM)：内存容量为 64K 字节。

③ 只读存储器(ROM)：共 32K 字节，固化有系统监控程序，BASIC 解释程序，LOGO 子集汉字管理程序及码表。

④ 汉字点阵库：固化有全点阵的国标一，二级汉字字库，包括 6763 个汉字及外文字母等。

⑤ 显示器接口：一个是经过调制器的射频信号输出，可连接PAL制式彩色或黑白电视机；另一个是全电视视频信号输出，可接标准的PAL制式彩色或单色监视器。

⑥ 盒式磁带录音机接口：盒式磁带录音机接口可与一般家用录音机相连接。提供输出信号电压是25mV，输出阻抗100Ω，要求输入信号峰值电压为1V，输入阻抗12kΩ。

⑦ 软盘驱动器接口：该接口可连接一台5.25英寸软盘驱动器。

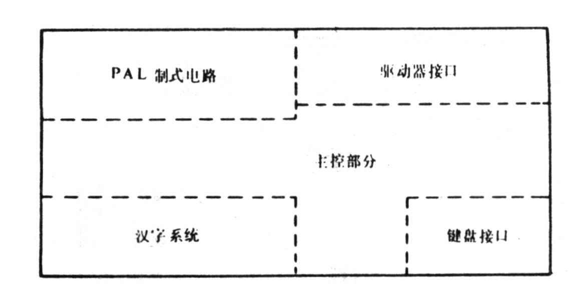


图1.5 CEC-I主机内部结构

⑧ 游戏操纵杆接口：9针的游戏杆插座，提供3个TTL电平开关量输入，4个模拟量输入。

⑨ 扬声器接口：内接0.25W，8Ω扬声器。

⑩ 键盘与键盘接口：键盘是标准的打字机键盘，具有69个键，采用弹簧键体。

⑪ 扩充槽口：在主机板上留有一个与APPLE II机兼容的50线输入/输出插座。此插座可插打印机接口电路或其它扩充电路板。

⑫ 电源：采用开关电源，最大交流功耗为25W。

1.3 键盘和屏幕显示

人与计算机的信息交换是通过键盘和屏幕来完成的。利用键盘可向计算机输入命令和程序，而计算机则通过屏幕显示命令或程序执行的结果，并且还会显示主机对用户所提出的要求和命令等。因此这两部分是人机对话最重要的工具之一。

中华学习机的键盘安装在主机外壳上，具备标准打字机键盘的各种键，并按照标准打字机的键位排列。另外中华学习机键盘还增加了一些功能键，中西文转换键，光标键，中断键等。

中华学习机屏幕显示可以用一般的家用彩色电视机或黑白电视机，也可采用彩色监视器或单色监视器，并且电视机和监视器可以同时使用。但监视器产生的图像比电视机图像要清晰些，并且稳定不闪烁，操作时不会引起人眼的疲劳。

标准的显示屏幕有三种不同的显示模式：一种是显示文字字符的文本模式，另外两种是低分辩率图形和高分辩率图形模式。在文本模式中，屏幕的显示功能为：

① 西文方式：字符构成为 5 × 7 点阵，每帧字符为 40 × 24。

② 中文方式：字符构成为 16 × 16 点阵（汉字），8 × 16 点阵（ASCII），每帧字符 17 × 10（汉字），34 × 10（ASCII 码），另外加一个状态行。

图形模式的屏幕显示功能为

① 低分辩率彩色图形显示：分辩率为 40H × 48V，16 种颜色。

② 高分辩率彩色图形显示：分辩率为 280H × 192V，6 种颜色。

1.4 盒式磁带录音机

由于 CEC－Ⅰ中华学习机的内存容量不是很大，只有 64K 字节，而且一旦关掉电源，内存 RAM 的内容就会消失，所以常常用盒式磁带录音机来延伸 RAM 的容量。也就是说，我们可以把程序存放到录音机的磁带上，以备日后再拿出来使用（将磁带上的程序调入到内存）。

录音机可选择专用的磁带机也可以选择家用录音机，但要求录音机的质量上能够保证走带平稳，频率响应好，不失真。录音机的磁带也可选用普通录音磁带。

1.5 软盘驱动器

作为外存储器，软盘驱动器的功能远远胜过盒式磁带录音机。这是因为磁盘存放程序比磁带更可靠，存储量更大，而且软盘驱动器的启动速度快，可随机存取文件。

中华学习机使用的是 5.25 英寸软盘驱动器，采用单面单密度记录方式，格式化以后的容量为 143K 字节。与此相对应，软磁盘也应选用 5.25 英寸单面单密度磁盘，或双面双密度磁盘。

1.6 打印机

利用打印机可以把我们所需要的文件，程序，报表，甚至图形打印出来。使用打印机必须要由打印机接口电路板与中华学习机相连才能使用。打印机接口电路板是通过插在中华学习机上的扩充槽口与系统相连。

中华学习机可配置多种类型的点阵式 Centronic 标准接口打印机，如 CP-80，RX-80，MX- 80，FX- 80 等。随机软件已提供打印 4 种汉字字形。

1.7 系统软件

CEC- I 中华学习机的主机上已固化有系统监控程序，CEC- BASIC 语言，LOGO 语言，汉字管理程序以及小汇编程序。这些软件开机后就可以直接使用。另外由于 CEC- I 与 APPLE II 系列微机兼容，因此 APPLE II 系列的各种语言软件及软件包均可使用。如 DOS 3.3，LISA2.5 等。

1.8 应用软件

CEC- I 是家庭教育和中小学计算机辅助教育的有力工具。目前已为 CEC- I 开发出了一系列的教育软件，其中的辅助教学软件包括从学龄前至大学的数学，物理，化学、语文、英语、生物、地理、历史等科目。

CEC- I 还配有适用于幼儿教育，青少年科技启蒙和智力开发，以及成年人尤其是老年人娱乐用的棋牌类等各方面的智力游戏软件。这些软件知识性强，趣味性强，富有想象力，启发创造性，特别适合青少年课外活动和家庭娱乐活动使用。此外，还有家庭管理软件，如家庭财务，备忘录，通讯录，保健指南，家庭医生，心理卫生咨询，旅游指南，菜食制作，电脑秘书等。

第二章
CEC–I 中华学习机的操作使用

对于任何一个电子计算机系统，当你第一次使用它时，尽管它的电路连线已经全部接好，但往往会有一种畏惧心理。在使用 CEC– I 系统时，同样也会有这种感觉。在这一章里，我们将讲述如何顺利地使用 CEC– I 中华学习机，相信你在学完这一章之后，一定会对 CEC– I 系统的使用与操作有充分的了解，为以后的学习建立起信心，使 CEC– I 中华学习机成为你真正的朋友。

2.1 CEC–I 系统的安装

2.1.1 开箱

将装有中华学习机的纸箱放在桌面上，正面朝上打开纸箱的箱盖，你会发现主机装在塑料袋中，两头各有一块泡沫衬垫。

小心取出主机，并清点随机带的视频电缆，录音机电缆，使用手册等附件。要妥善保存好纸箱及其它物品，以备将来搬运时使用。

主机箱是一个紧凑的塑料机盒，它的右上角有一个电源指示灯(小孔内)，电源一接通，此灯便亮了。主机的正面是键盘，左侧面是一个电源开关，后侧面有电视机接口，显示器接口及软盘驱动器接口，而左侧面有一个盒式磁带录音机接口及游戏操纵杆接口。如图 2.1 所示。

2.1.2 显示器的连接

CEC– I 的显示器可以是电视机(彩色或黑白)也可以是监视器(彩色或黑白)，但连接的方法不同。

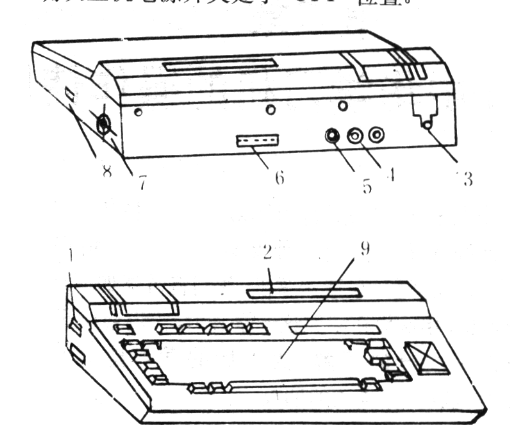
①电视机连接(参看图 2.2)

· 把电视机的天线插头拨下。

· 将视频电缆线的一头插入主机上的电视机接口，另一头插入电

视机的天线插座上，电视机天线插座的输入阻抗为 75 Ω 。

确认主机电源开关处于“OFF”位置。



1.电源开关，2.扩充槽口，3.电源线，4.电视机接口，5.监视器接口，6.磁盘驱器接口，7.录音机接口，8.游戏杆接口 9.键盘

图 2.1 主机接口

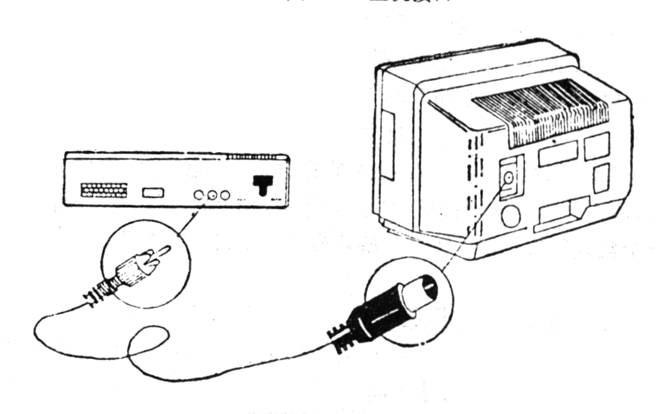


图 2.2 电视的连接

· 将电视机电源插头和主机电源插头分别插入 220V 电源插座上.

· 把电视机音量开关调到最小位置。

· 先打开电视机电源开关，然后再打开主机的电源开关。

· 当使用黑白电视机时，首先调准电视机频道(生产厂家要求的频道为准，一般为 3 频道)。开机后，调节频道微调及对比度，亮度旋钮，直到电视屏幕上出现下面几行清晰字样为止:

ZHONG HUA XUE XI JI

VERSION 1.1

当使用彩色电视机时，同样首先选好频道(3 频道)，并通过自检方式进入显示彩条图象(参考 2.4 节机器的自检)，然后调节频道微调及色饱和度，亮度，对比度，直到屏幕上的彩条满意为止。以后可把这一频道作为计算机专用频道，再开机就可以不用调机了。

②监视器的连接(参阅图 2.3)

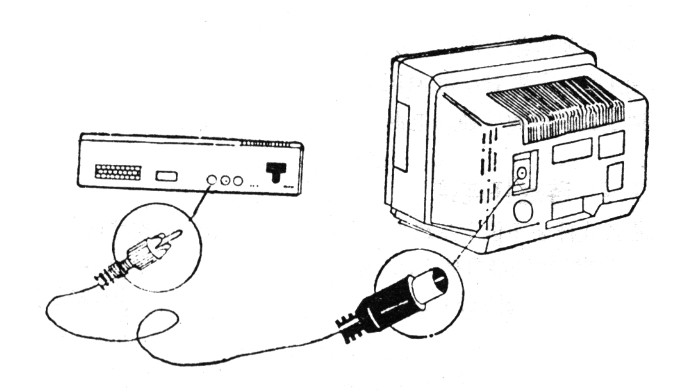


图 2.3 监视器的连接

· 将视频电缆线的一头插入主机监视器接口，另一头插入监视器的“IN”插座。

· 确认主机电源为“OFF”位置。

· 将主机电源及监视器电源插头分别插入电源。 先打开监视器电源开关，然后再打开主机电源开关，屏幕上将出现清晰的“ZHONG HUA XUE XI JI”字样。

2.1.3 录音机连接(参阅图 2.4)

· 确认主机电源为“OFF”位置。

· 把录音机电缆线的一端(五芯插头)插入主机右侧面的录音机接口。

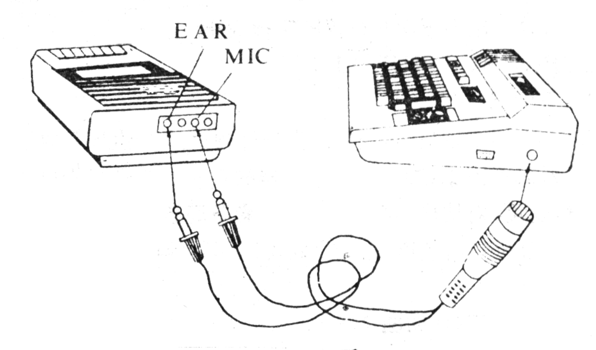


图 2.4 录音机连接

· 将录音机电缆线的另一端的两个 3.5mm 插头分别插入录音机 EAR 插孔(连接主机 IN)和 MIC 插孔(连接主机 OUT).

· 把录音机电源线插入电源。

2.1.4 游戏杆连接

九芯游戏杆插座位于主机右侧(如图 2.5 所示),把连接游戏杆的插头直接对准游戏杆接口插入,必须注意:两者的信号一定要一一对应(游戏杆接口信号如图 2.5)。

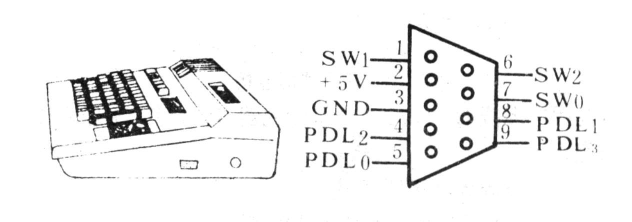


图 2.5 游戏杆的连接

2.1.5 磁盘驱动器连接(参阅图 2.6)

· 确认主机电源开关处于“OFF”位置。

驱动器接口上。必须注意的是插针与插孔要一一对应，电缆线插头的凸出部分朝上，否则将会损坏软盘驱动器。

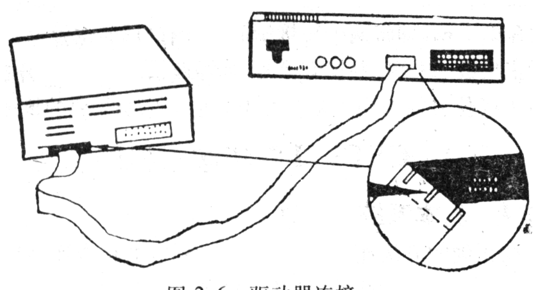


图 2.6 驱动器连接

2.1.6 扩充卡的连接(参阅图 2.7)

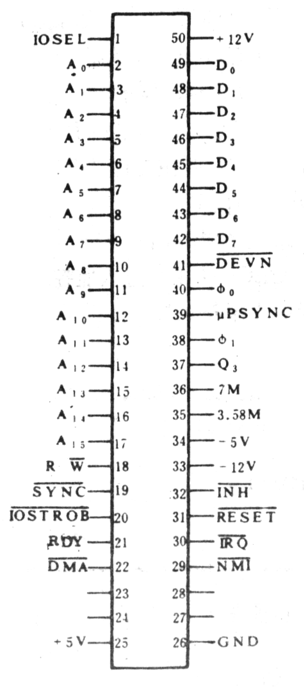


图 2.7 扩充卡的连接

CEC－I 主机板的右上侧有一个 50 线的扩充槽，可插入多种外设卡及扩充卡(如打印机卡，音乐卡，A/D 卡等)。扩充卡如果尺寸太大插不进去，可采用 50 线转接板把槽口抬高，再插外设卡或扩充卡。具体步骤如下：

确认主机电源开关在“OFF”位置。

用手轻轻压下主机右上方的扩充槽盖，可看到印刷板上安装了一个 50 线插槽。

把扩充卡按元器件朝前，焊接面朝后的方向对准槽口往下插。必须注意，一定要平插插平，以免错位。

扩充槽号的选定

用户可以根据需要用短接帽来选择槽号，槽号可以为 1，2，3，4，5，7

中的任一个。由于 CEC- Ⅰ在设计上已将 3 号槽定于汉字系统，6 号槽定于磁盘驱动器接口，所以用户不可选用。CEC- Ⅰ在出厂时一般都将槽号设在第 1 槽，即将两个短接帽插在标有“1”的位置上。选择槽号插头位于主机板的右侧，并标有槽号，如需改槽号，需要打开主机机壳，把原在第 1 槽位置的两个短接帽取下并插入你所需要的槽号位置上。

2.2 CEC- Ⅰ中华学习机的启动

2.2.1 基本系统的启动

首先按正确的方法连接好 CEC- Ⅰ系统，并检查连接是否都正确无误。然后，将显示器的电源打开，调好对比度，如果是电视机还应把频道调整好(一般用 3 频道)。接下来就可以打开主机电源了。

当开机之后，立刻就能听到机器发出“嘟”的一声。这声音告诉你：主机已准备就绪，同时主机右上方的电源指示灯应该发亮。如果一切都正常。CEC- Ⅰ将自动进入西文 BASIC 状态，在屏幕上将会显示如图 2.8 所示的文字，此时你就可以输入命令或 BASIC 程序了。

当机器启动以后，在屏幕上可以看见一个十分明显的，有规则的，忽亮忽暗的闪烁小方块，这个闪烁的白色小方块叫做光标，它的位置表示了你从键盘上打入的字符在屏幕上出现的位置。

在光标的左边一个“]”符号，称之为提示符。符号“]”是 CEC- BASIC 语言系统提示符。另外 CEC- Ⅰ中当进入监控程序后的揭示符为“*”，当进入小汇编程序后的提示符为“!”。

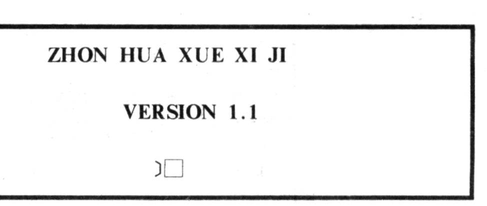


图 2.8 机器启动后的屏幕显示

2.2.2 连接驱动器系统的启动

首先按正确的方法连接好系统。在启动前打开磁盘驱动正面中央的盖子，把含有 DOS3.3 磁盘操作系统的磁盘印字面朝上，水平而轻轻地沿着驱动器的细缝插入驱动器内。当磁盘完全插入以后，再关上驱动器的盖子。然后就可按照基本系统的启动方法进行启动。当主机电源打开后，屏幕上首先显示如图 2.8 的文字幕，然后磁盘驱动器上的指示灯发亮，并且可以听到驱动器马哒的转动声音，这时机器正在把 DOS3.3 磁盘操作系统载入内存。当磁盘操作系统载入内存以后，驱动器指示灯熄灭，屏幕上将会出现 DOS3.3 磁盘操作系统的题头版面，如图 2.9 所示。

```
DOS VERSION    3.3 08/25/80
APPLE IIPLUS OR ROMCARD SYSTEM MASTER
(LOADING INTEGER INTO LANGUAGE CARD)
]□
```

图 2.9　DOS3.3 题头版面

此时 CEC- I 就已在 DOS3.3 的控制之下接受用户的命令和 BASIC 程序。

如果开机前没有将驱动器盖合上或没有将含 DOS3.3 的磁盘插入驱动器，CEC- I 就会直接进入 CEC- BASIC 语言系统，而不受 DOS 3.3 的控制。以后如果需要装入 DOS 3.3 系统，可以将 DOS 3.3 磁盘插入驱动器合上盖，并键入

PR#6(回车)

于是 DOS3.3 将被载入机器内存。

2.3　键盘操作

CEC- I 的键盘与主机之间通过一条 26 线扁平电缆线连接，直接装入主机机壳内。此键盘具有重复功能和 N 键滚动功能。

所谓重复功能是指当某键按下时间大于一定值时，连续不断地重复发送这个键值，即自动重复此键。所谓 N 键滚动则是指在有多个键按下时，将根据按下键的先后次序分别检测并输入主机，同时也显示在屏幕上。

图 2.10 是 CEC－Ⅰ键盘的排列图。在这个键盘上，有些键上标有0,1,2,3…9以及A,B,C,D…X,Y,Z等字符，这是我们大家十分熟悉的数字和英文字母，使用也很方便。例如，当你按下“4”这个键，屏幕上就会出现一个“4”。

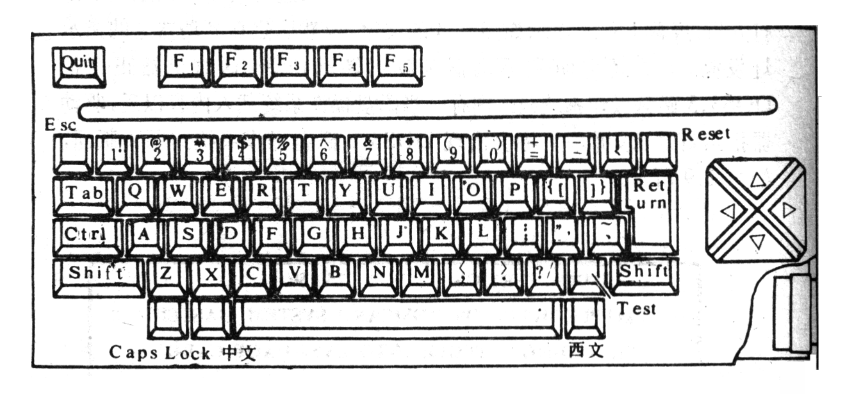


图 2.10　键盘排列图

除了上述的 36 个数字键和字母键之外，还有.、:、—、,、;、/ 这 6 个键的用法也与上面的 36 个键用法相同。另外最下面的那个长键，称为空格键。按一下空格键，显示屏幕的光标就向右移一格，字符显示出现一个空格，即这个位置什么也不显示。

还有一些键具有特殊的作用，称为功能键。下面我们分别给以说明。

2.3.1　SHIFT 键

这个键共有两个，分别安排在键盘的左侧与右侧，这是为了操作起来方便。这个键的功能是当按此键和另一键时，产生上档键码。例如，在标有数字“5”的这个键。其“5”的上方还标有“%”。如果要把“%”这个字符显示到屏幕上，就必须将 SHIFT 与“5”这两个键同时按下，或者先按着 SHIFT 不动，然后再按一下“5”。同样，凡是键上有上下两个字符的，要想使用上面的那个符号的话，必须借助于 SHIFT 键。

2.3.2　CAPS LOCK 键

CAPS LOCK 是英文 CAPITALS LOCK 的缩写,意思是英文“大写字母锁定”。当按下此键后,从键盘上输入的 26 个英文字母将是大写字母;当抬起此锁定键以后,从键盘上再输入的 26 个英文字母为小写字母。

2.3.3 Ctrl 键

Ctrl 是英文 Control 的缩写,是“控制”的意思。Ctrl 键总是与另外一个键同时使用,即按住 Ctrl 键不放,再按另一键,使得同一键具有另外的功能。例如,当“G”键与 Ctrl 键同时使用时,“G”键就显示出它们合用的特有功能:发出高频笛声“嘟”的一声。Ctrl 键还有其它不同的组合,可引起 CEC-I 机不同的反应,目前我们暂时不去详细讲述,以后会逐步提到。

2.3.4 ◁ 和 ▷ 键

这两个箭头键叫做左箭号键和右箭号键。利用这两个键可以修改输入过程中的各种错误。◁ 键象打字机上的空格退位键一样,每按一次,光标退回一格,退回光标下的字符就从 CEC- I 的内存中抹去。现在,不妨在键盘上试一试,首先,打入一串任意字符(例如 PRINT),然后,连续几次按下 ◁ 键,在屏幕上就会看到光标沿着刚打入的字符退位返回,但原来字符在屏幕上并不消失,然而,这些字符实际上已从 CEC-I 的存储器中抹去了。当光标退位返回到达屏幕左边沿时,若再继续按下 ◁ 键,则光标就跳到下一行,于是,产生一个新的提示符。

▷ 键是使光标沿显示行向右移动(前进方向),当按下此键时,光标沿着显示行向前移动,它所经过的每一个字符,又重新复制到存储器里,就象重新按下了某个键来得到一个新的字符一样。为了更清楚地了解 ▷ 键的作用,不妨在键盘上连续地打入一串字符,并用 ◁ 键将光标返回几格,让它从存储器中抹去,然后再按几次 ▷ 键,每按一次这个键,光标经过的字符就重新准确地又送入了存储器,如同重新打入了一样.

2.3.5 — 和 ▽ 键

这两个键叫做上箭号键和下箭号键。利用这两键与◁ 和 ▷ 键配合就可以很方便地修改编辑屏幕上任意位置上的字符,以及重复执行一条

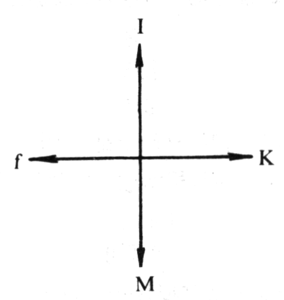
已经执行过的命令。按△键将使光标上移一行，按▽键使光标下移一行。例如，我们原先从键盘打入了下面一条命令：

]PRINT“ABCDEFG”

当执行完这条命令以后，如果我们还想执行　PRINT “ABCD” 命令。那么，只要利用△键，把光标上移到上面执行过的命令行，再利用▷键扫描这一行的命令字符到“E”，在此处打入一个双引号，回车以后就可以执行这条命令了。

2.3.6 Esc 键

Esc 是英文 ESCAPE 的缩写，意思是”脱离”。这个键的用法比较特殊，值得注意的是这个键只要按一下，放掉后仍会起作用，而不象 Ctrl 和 Shift 键那样要一直按着，直到另一个键按下后才会起作用。按一下 Esc 键，就是通知 CEC- I 进入“脱离”状态。什么叫”脱离”状态呢?让我们来试一下，先按一下 Esc 键，然后再按几次”K”键，这时屏幕上并不会显示”K”字符，而是使光标向右移动几次，然后再按几次”M”键，这时光标向下移动，再按”I”键，则光标向上移动，最后再按”J”键，会看到光标向左移动。按了这 4 个键以后，会发现这 4 个键”K、M、I、J”在键盘上正好形成十字，分别代表 4 个方向，即：



只有 CEC-I 在”脱离”状态时，这 4 个键可以反复地用来改变屏幕上光标的位置。若要使 CEC-I 从”脱离”状态回到原来状态，只要按一下空格键或其它任何一个键即可。这就是 Esc 键的功能，它可以任意地将光标移到屏幕上的各种位置上。

Esc 键在调试程序时非常有用。比如，在调试列出程序时，当发现前面程序有错，而光标此时已在下面时，只要按一下 Esc 键，然后利用

I、K、J 键,可把光标移到需要修改语句编号(称行号)的第一个字上。此时先按一下空格键或任何一个键,把程序回到原来状态。接着,按 键,把光标移到需修改的字符上即可,键入需要的字符之后,按 键向后移,一直移到需要修改的这一条语句末尾的后一个位置,然后按"回车"键即可。

2.3.7 Return 键

这个键称为"回车"键,它是一条命令的执行键。从键盘上打入一串字符后,按下 Return 键,表示输入的内容已完,请机器执行。机器就马上对刚才送给它的一串字符进行阅读分析并执行之。实际上,当你用按键的方法,将要告诉机器的语句或命令逐个送到屏幕上之后,如果不按 Return 键,这些字符只能停留在屏幕上,而不会到达机器的内存中去,机器等于没有看到你输入的内容。而按下 Return 键后,屏幕上的内容马上就被搬入内存,机器就可立即执行。因而,在输入一串字符到显示屏幕上时,如果发现内容或按键有错误的地方,只要不按 Return 键,就可以从容不迫地修改,直到全部正确后,再按 Return 键。在按下该键之后,再想修改就得采用前面提到的 Esc 键与修改方法。

2.3.8 Reset 键

这个键称为"复位"键或"中断"键。当你的程序在执行过程中,机器失去控制,或者程序在执行了一半不希望它继续执行下去了。此时一种办法就是关掉机器电源,不过这样就会失去内存中的程序和数据,另一种办法就是利用 Reset 键。当需要进行系统复位时,必须在按下 Ctrl 键的同时再按下 Reset 键,这样释放以后就会中断 CEC- I 的任何工作,系统便重新启动进入 BASIC。并把控制权交还给键盘。

2.3.9 其它一些功能键

Test 键: 当同时按下 Ctrl-Reset-Test 这三个键时,CEC-I 主机进行自检测。

Quit 键: 从监控返回 BASIC 状态,它不破坏内存里原有的 BASIC 程序。它相当于按 Ctrl- C 键。

Tab 键: 象是打印机上的定位键,好比说屏幕定了 1,11,21,31 四

个位置，光标在 13 那儿，按 Tab 键就会移到下一个定位 21。

F1 键: 进入 ASCII 方式(中文状态的字母方式)。

F2 键: 进入中文状态的拼音方式。

F3 键: 进入中文状态的区位方式。

F4 键: 供用户自定义使用(键码为 14)。

F5 键: 供用户自定义使用(键码为 06)。

中文键: 进入中文状态。

西文键: 进入西文状态。

以上这些键的用法我们将在后面的有关章节详细说明。

2.4 CEC-I 主机的自检测

当机器启动后，同时按下 Ctrl-Reset-Test 这三个键，释放以后，就可以对主机的内存单元(RAM，ROM)以及彩色显示进行自检。在自检时，屏幕上将显示相应内存单元的检测内容和检测结果。如果机器工作正常，应显示如图 2.11 的结果。

```
            MEMORY-TEST

                    TIMES: 0001

   RAM      BFFF    OK
   BNK1     FFFF    OK
   BNK2     FFFF    OK
   ROM1     BFFF    LOGO
   ROM2     FFFF    CEC-BASIC
   AUX1     BFFF    HZTABLE
   AUX2     FFFF    HZPROGRAM
   AUX3     5D99    CECWL
```

图 2.11 自检内容和结果

其中 RAM 部分是检测地址 $ 0000 ~ $ BFFF 共 48K RAM。

BNK1 和 BNK2 部分是用两个软开关测试地址 $ D000 ~ $ FFFF RAM。

以上三个内存单元当测试正确时，屏幕相应处示"OK"；如有出错，屏幕相应处将显示"ERR"。

ROM1 是测试 LOGO 语言的。

ROM2 是测试中华学习机 CEC-BASIC 解释程序和系统监控程序的内容。

AUX1 测试汉字码表内容。

AUX2 测试汉字管理系统内容。

AUX3 测试 1 兆位的汉字点阵库内容。

以上五个单元当测试正确，屏幕将显示相应的测试内容名称；如果测试出不为所知的内容，屏幕将相应地显示"UNKNOWN"，当没有加汉字库时，屏幕将在相应位置显示"NULL"。

当 RAM，ROM 测试完后，将进行彩色显示测试，屏幕将显示 16 种不同颜色的彩条。

自检程序可以连续循环地测试，并在 TIMES 一栏显示测试的次数。如需调试彩色，可在屏幕显示彩条时按任意键使屏幕一直显示彩条，直到你调试完彩色电视机，再按任一键又进入自检程序循环测试。

如果要中断自检测试，可同时按下 Ctrl- Reset 键，机器便进入 CEC- BASIC 状态。

2.5 汉字输入方法

当机器启动以后，屏幕显示方式为西文文本方式，此时若敲入中文键，系统将进入中文显示方式，屏幕显示为高分辩率图形方式。进入中文方式以后，屏幕可以显示 10 行汉字，每行 17 个汉字。屏幕底部的第 11 行为状态行，状态行最左端显示"字母"两字，表示系统处于中文方式的字符(ASCII)输入状态。此时若按下 F2 键或 F3 键，状态行左端将显示"拼音"或"区位"两字，表示系统进入了中文方式的拼音输入状态或区位码输入状态，在这两种状态返回到字符输入状态，可按 F1 键。另外，在进入中文方式以后，如果同时按下 Ctrl- O 这两个键，可以控制状态行的显示与不显示，但不改变输入方式。

除了键盘方式进入汉字显示方式之外，程序方式也可以进入汉字

显示方式。在 BASIC 程序中，使用 PR#3，也可以进入汉字显示方式。

在中文方式下，若按下西文键，将退出中文显示方式，屏幕显示将返回到西文文本方式（低分辩字符方式）。在 BASIC 程序中使用 TEXT 或 PRINT CHR $ (17)，也可以退出汉字显示方式。

2.5.1 拼音输入方式

在进入汉字显示方式后，按下 F2 键或 Ctrl−L 键，屏幕底部状态行将显示“拼音”两字，此时即进入拼音输入方式。

CEC−I 中华学习机的拼音输入为全拼音输入方式，即输入汉字时将该汉字的音节的声母和韵母全部敲入。计算机根据输入的音节在状态行提出该音节的所有字，其中包括了该音节所有不同声调的字。操作者可以根据状态行上提示的汉字序号用数字键选择所需要的汉字。因此，只要学过汉语拼音的人就很容易学会全拼音输入方法。下面是全拼音输入汉字的方法。

① 第一键提示常用字：当你第一键敲入了汉字的声母以后，屏幕状态行上将显示出该声母开关音节中的最常用的 6 个汉字。此时若有需要的汉字，只要敲入相应汉字的序号即可；若没有，则要继续敲入该汉字的韵母。

② 向后或向前寻找同音节字：当敲完汉字的韵母后，状态行提示出 6 个同音节字。若出现所需要的汉字，即可敲入相应的汉字序号；若这个 6 个同音节字中没有要输入的汉字，则可按 > 键继续向后寻找。每按一次 > 键，屏幕状态行就显示出下一幕 6 个同音节字，直到机器发出“嘟“的一声，表示该音节的汉字已全部显示完。当要寻找前一幕汉字时，可按下 < 键，此时，前一幕的 6 个汉字以显示在状态行上。

③ 敲入所选汉字的序号：当屏幕状态行出现要输入的汉字时，只要敲入该汉字前面的序号，该汉字就会显示在屏幕当前光标的位置上，且该汉字的内码被输入到键盘输入缓冲区。连续敲入序号，可连续输入汉字。

④ 删除拼音提示：若输入汉字时，有拼音敲错，可直接按 ◁ 键，使状态行上最后一个拼音字母被删除，或者按空格键，使所有提示字符都删除。当状态行上的拼音字母都删掉以后，再按 ◁ 键，则删除光标上的

信息。

2.5.2 区位码输入方式

当进入汉字显示方式后，按下 F3 键，屏幕底部状态行将显示出“区位”两字，此时即进入区位码输入方式。

进入区位输入方式以后，将使用国家 GB- 80 的区位方法输入汉字。在输入汉字时，仅使用 10 个数字键即 0 ～ 9，每 4 位数字对应一个汉字。例如，国标中标定 16 区 01 位置的汉字是“啊”，键入 1601 后，“啊”字就显示在当前光标的位置上。其中前两位数字表示区号，后两位数字表示位号，国标中一级汉字库都在 55 区以前，二级汉字库是从 56 区到 87 区。因此我们可以通过查阅区位码表寻找出汉字的区位码，来输入汉字。利用区位码输入汉字的优点是输入速度快，没有重码，即每输入 4 个数字就可以输入一个汉字。另外利用区位码输入还可以输入一些特殊的字母和符号。

当敲错区位码后，可按△键，使区位码提示的最后一个数字被删除，当区位码都删掉后，再按△键，则删除光标位置上的信息。

2.5.3 特殊符号的输入

在拼音方式下，当按下 = ，± 或 ! 键以后，可以选择输入标点符号，算术运算符号和制表符号。但这些符号在程序中，只能作为字符处理，不能参加运算。例如，在拼音方式下按　键，将可以选择输入下面一些符号：$、%、+、−、×、÷、±、=、≈、>、<、⩾、⩽、/、(、〔、)、〕、∑、～、①、②、③、④、⑤、⑥、⑦、⑧、⑨、⑩。

2.6 磁带录音机的使用

人们把一个程序从键盘上打入计算机，待运行和工作完成后，通常随手就把电源关掉了。在关掉电源的同时，键入计算机内存中的程序也随即消失。再想运行这段程序，又得一句一句地从键盘打入，这是十分麻烦的事。为了解决这个问题，就必须采用录音磁带（或软磁盘）。在还没有关机之前，先把程序存储到磁带（或磁盘）上，然后就可以放心地把计算机的电源关掉。下次要用这段程序的话，只要发一个装载命令，再从磁带（或磁盘）上取回来就可以了，既方便又可靠。CEC- I 在设计

时已经备有专门的录音机接口，供存放和读取磁带程序使用。

2.6.1 盒式磁带的处理

使用盒式磁带时应十分小心，它容易损坏，而且一经损坏就不易还原。使用时，注意不要触摸盒式磁带内的磁带表面，要保持清洁，不要让油污浸渍磁带。为了确保磁带不受损坏，不用时，应将它放回磁带盒内保存，切勿把它放在灼热的地方，或者让阳光直接照射，也不能放在磁场附近，如不能放在电机附近。

2.6.2 音量的调节

在 CEC-I 装入程序的过程中，须将录音机的音量调节到正确的电平。如果音量太高或太低，读入信息会发生畸变，CEC-I 便无法正确识别。

对于所用的某一录音机，只能用累试法来确定一个正确的放音音量。一般的过程是：在试装一个程序时，先将音量置于较低的位置，如果不能工作，将音量稍调大一点再试，反复几次，直至成功地装入程序为止。另外有些录音机音调控制也对信息的正确读入有影响。

2.6.3 游戏磁带软件的使用方法

① 将录音机音量调到适当位置(一般在中间到最大位置之间)，打开主机电源使机器进入 BASIC 状态。

② 把游戏磁带软件装入录音机，并倒带到头。

③ 从主机键盘上键入“PLAY”命令(先不要按回车)。

④ 按下录音机 PLAY 键，等到磁带走稳后(大约为 3 秒钟)，再按主机上的“回车”键。

⑤ 这时屏幕上应显示

WAITING......

再等待十秒钟左右机器将发出“嘟”的一声，并且屏幕上将显示游戏软件的中文标题。

接下来机器将把游戏程序读入内存。在读带过程中，屏幕的下方将不断显示一个个彩色小点，表示当前读入的数据正确。当全部程序读入内存以后 CEC-I 会自动执行游戏程序，于是你就可以按照游戏的操

作方法玩了。

下面是一些在读带过程中常出现的错误。

① 在读带过程中不出现中文标题，机器好象“死”在那里。这可能是录音机音量太小或太大，应适当调节音量。或者是磁带开始没有倒带到头，或者是按“回车”键太早或太晚。

② 在读带过程中，若读入的数据不正确，CEC-I 将用中文提示“读带出错”，这时应适当调节录音机音量或者是音调旋钮，并重新读带。

2.6.4 BASIC 程序的存取方法

(1)将 BASIC 程序存储到磁带上

①首先把要保存的程序从键盘上输入到内存。

②将一盒空白磁带放入录音机。一般录音机上均有计数器，应记下每次写带前计数器的读数。

③从键盘上打入"SAVE"命令(先不要按回车键)。

④将录音机置于"录音"状态，即同时按下录音机的 PLAY 和 REC 两个键。

⑤按下回车键，屏幕上的光标马上消失，大约过 10 秒至 15 秒后，CEC-Ⅰ会发出"嘟"的一声，即通知你"录音"已正式开始。前十几秒钟的空转是引导信号。录完后，CEC-Ⅰ再次发生“嘟”的一声屏幕上的光标又重新出现。

⑥按下录音机的 STOP 键，并记下计数器的读数。

⑦从录音机内取出磁带，在磁带盒标签上写上这一段程序的文件名以及计数器起始和结束位置，以备查用。

注：CEC-Ⅰ中华学习机存储 BASIC 磁带程序时，为程序定义一个文件名，这样就为磁带程序的管理带来很大的方便。带文件名的写带命令格式如下：

SAVE"文件名"

其中程序的文件名必须用双引号括起来，否则 CEC-I 将认为是错误的命令(不含 DOS3.3 时)。

(2)把磁带上的 BASIC 程序取到内存中

这个过程正好与上面讲的相反，它会将原先存放在磁带上的程序一字不漏地送回到内存中去。

① 将磁带放到录音机中去，并倒带到程序录制时的起始位置(可由计数器上获知)。

② 从键盘上打入"LOAD"命令(先不要按回车键)。

③ 录音机音量与音调调到适当位置，按下 PLAY 键。

④ 按下主机回车键，这时屏幕上光标消失，CEC－Ⅰ先接受程序的引导信号，当磁带进入数据区时，机器发生"嘟"的一声，表明开始正式将入程序。在程序读取完后，机器又会发出"嘟"的一声，屏幕上出现光标，表示完成了整个程序的装入。

⑤ 按下录音机 STOP 键，关掉录音机。

注：如果程序在存带时定义了文件名，则装载程序的命令应改为：

LOAD"文件名"

其中文件名要用双引号括起来，这样在装载程序的过程中，屏幕上会显示被装入程序的文件名。当装入的不是所要的程序时，屏幕会提示已装入的文件名和一个反相显示的字符"N"，并继续装入后面的程序，直到装入 LOAD 命令中所给出文件名的程序为止。如果 LOAD 命令不给出文件名，那么 CEC－Ⅰ只将所遇到的第一个程序装入内存。

在读带的过程中，如果没有听到"嘟"的一声，或者得到一个错误信息，那么应该重新检查录音机的音量控制。

当把欲装入的程序装入内存以后，用命令

RUN

便可以启动程序运行。

2.7 软盘驱动器的使用

磁盘驱动器是利用磁性材料的磁化特性来存储信息的，这一点与磁带录音机相同。二者的最大差别是：磁盘是圆形的，外形象唱片，也象唱片一样地转动。磁盘驱动器里面有个磁头，用来读出和写入信息。计算机可以把磁头移到磁盘表面的任何地方这种能力称为随机存取。因此，磁盘驱动器是一种随机存取的存储装置。它比磁带机的顺序存取方式速度要快得多，也准确的多。

CEC- I 中华学习机所具有的磁盘驱动器接口控制电路只支持 5.25 英寸单面单密度软盘驱动器，并且其接口控制信号与 APPLE DISK II 接口控制信号相兼容。

2.7.1 软磁盘的使用

软磁盘也称软盘或盘片。有多种型号和规格，它适用于不同类型的软盘驱动器。按照尺寸的大小分为：8 英寸、5.25 英寸及 3.25 英寸；按驱动器磁头工作方式可以单面和双面读写，按照数据的记录方式又有单密度和双密度两种。对于 CEC-I 来说，由于选用的是 5.25 英寸单面单密度软盘驱动器，因此，它对软盘的选择要求比较低。一般来说，单面单密度（SS／SD），单面双密度（SS／DD），双面单密度（DS／SD）以及双面双密度（DS／DD）的 5.25 英寸软磁盘均适合其使用。

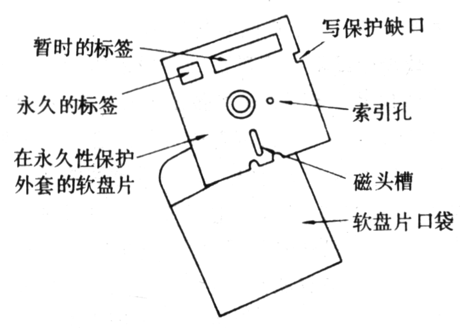


图 2.12 软磁盘的包装与外形

软盘密封于永久性保护套内（参见图 2.12），盘片由圆形脂塑料做成，其表面涂有磁性材料。使用时，软盘在保护套面旋转，磁头通过保护套中的长孔（又称磁头读／ 写窗口）和软磁盘表面接触，读写数据。

永久性保护套：由防静电皮革做成，内壁夹有防静电防潮材料，其作用是防止灰尘和保护盘面磁层不受损坏，同时也防止盘片旋转时所产生的静电而使信息丢失。

索引孔：用来进行盘片划分扇区的物理定位标志。由于大部分软盘驱动器上没有装上索引孔识别装置，故索引孔不起作用。

磁头读写窗口：这是在永久性保护套上开辟的一个长孔，是磁头与

软盘接触，读写信息的。因此，在软盘使用过程上，绝对禁止用手指或异物接触这个长孔中所暴露出的磁盘部分。

写保护切口：控制着软盘驱动器写电路开关，当用写保护封条贴在该切口上时，可以防止磁盘被写入任何信息。

永久性标签：出厂时贴上的磁盘说明。用以指明磁盘的牌号和型号等。在使用磁盘时，最好用手指捏着这部分取盘和插盘。

临时标签：在使用软盘，我们希望能够通过简单的方法注明该盘片的名称、作用等。此时我们可以用不干胶标签贴上，用软笔写上名称、标记等。不需要时，可以撕下该标签，换贴新的标签。

盘纸套：用来保护盘片，防止灰尘的侵蚀。我们要养成把不同的盘片及时插入纸套中的习惯。

使用软盘时要小心地取放，手指头千万不要直接接触软盘上的薄膜，严禁脏物灰尘进入，也不能用重物压它，更不能把它放在发热体的附近以及磁性物体的旁边。不用时要用纸袋套好，放入软盘匣内。

把软盘插入软盘驱动器应将磁盘夹在拇指和食指之间，打开驱动器的机箱门，轻轻地几乎不受阻碍地将磁盘送入盒内。如果磁盘不是很顺利地滑进去，就把它取出来，再试着放进去。磁盘插入驱动器后，把门轻轻地关上，关门时也应感到轻松；如果门与磁盘有一点抵触，就要把门打开，把磁盘全部推进去后，再把门关上。使劲硬关，会损坏磁盘或驱动器。有时，也可等待驱动器转动后，再把门关上，因为驱动器转动后，会将磁盘自动地置于驱动器的中央位置。

CEC－I 为你提供了 DOS3.3 磁盘操作系统软件，用来对软盘上的文件进行管理。因此，在使用软盘来存取程序之前，必须先要把 DOS3.3 磁盘操作系统装入到内存中去，以后凡是要与软盘交换信息的动作和控制，都由 DOS 来管理。关于 DOS3.3 的引导方法请参考 2.2.2 一节中的介绍。

2.7.2 DOS3.3 磁盘操作系统简介

CEC－I 中华学习机具有丰富的系统软件和应用软件支持，在这众多的系统软件和应用软件中，用户接触最多，使用最广泛的是美国 APPLE 计算机公司为 APPLE II 微机系统设计开发的一种磁盘操作系统："DOS3.3 磁盘操作系统"。

磁盘操作系统是一组程序，它能自动地管理储存文件的磁道，能够完成许多内务任务，使磁盘驱动器能够顺利地运行。DOS3.3 磁盘操作系统除包括了一般磁盘操作系统所具有的基本功能(即磁盘数据的读 / 写，磁盘文件的建立和管理，输入 / 输出接口命令等)外，还专门为 BASIC 程序的使用提供了更多的操作命令。它与系统软件，用户程序的接口界面非常清晰，用户可以通过它所提供的一些命令很容易实现主机内存与磁盘之间的信息交换。

DOS3.3 磁盘操作系统是用 6502 机器指令编写的，它由三大部分组成: 主体程序，文件管理程序以及磁盘驱动程序。当系统引导后，将被装入主机内存，并且以后一直常驻内存。DOS3.3 操作系统程序占内存约 10K 左右(从 $ 9600 ~ $ BFFF)。

经过 DOS3.3 初始化操作后的软盘，被划分为 35 个磁道，每道划分为 16 个扇区，每个扇区可记录 256 个字节的信息，所以整张软盘可以储存 143K 信息。但是由于第 0, 1, 2, 这三个磁道上存放的是 DOS 操作系统的副本程序，第 17 道存放的是磁盘管理信息，而其它磁道才可用来存放用户程序，所以用户实际可用的盘区为 496 个扇区，共 124K 字节的存储空间。

在没有引导 DOS 操作系统的情况下，用户只能使用 ROM 中的 CEC- BASIC 解释程序，而在引导 DOS 操作系统后，便可在 BASIC 方式下调用 DOS 命令。DOS 操作系统命令由命令保留字和命令参数两部分组成。其命令格式如下:

COMMAND ANME〔, Ss〕〔, Dd〕〔, Vv〕

在上述 DOS 命令语法描述中，方括号之中的项为可选择命令参数，大写字母的参数标记，小写字母为参数。可选择命令参数的顺序是任意的。命令参数说明如下:

① NAME: 文件名，它由字母打头的字符串组成(其中不能含有逗号)，文件名的有效长度最多可达 30 个 ASCII 字符。

② Ss: 设定软盘驱动器所在的槽口号，s 值可以是 1 ~ 7 之间的整数(但不允许为 3)。

③ Dd: 设定软盘驱动器在接口卡上的设备号。

④ Vv: 软盘的卷号，一张软盘只能有一个卷号，卷号的值 v 可以是 1 ~ 254 之间的整数。

DOS 在启动引导之后的默认参数值为 s=6，d=1。

2.7.3 软盘的初始化处理

新买来的空白软盘片，上面没有记录任何信息，在使用 DOS 命令对空白软盘记录信息之前，必须对它进行格式化处理。格式化的目的是在磁盘上按磁道和扇区来划分地址，并把地址以数据的形式记录在软盘上，便于读 / 写软盘数据时，按地址进行查找。

DOS3.3 提供了一条初始化软盘的命令(INIT)，利用初始化操作命令，可以完成对软盘进行格式化的操作。

DOS 初始化命令的格式为:

INIT 文件名

其中文件名是由一个英文字母开头的一串字符组成，最多不能超过 30 个字符。一般我们把这个文件名所代表的程序称为“欢迎”程序。

使用 INIT 命令，可把现已在内存中的程序存到软盘中去，这个存入的程序就变成了“欢迎”程序。以后每次用这张软盘启动机器时，该程序都会自动运行。一个好的欢迎程序将会指明软盘的内容。典型的欢迎程序如下:

```
100  TEXT
200  CALL- 936
300  PRINT "THIS IS MY FIRST DISKETTE"
400  PRINT
500  PRINT "INITIALIZED 3/16/88"
600  PRINT
700  PRINT "BY GU XI-XIONG"
800  END
```

使用软盘时，若把盘中的欢迎程序删除掉，那么，每当由软盘启动时，就会看到出错信息 FILE NOT FOUND(文件找不到)。为了防止删除掉欢迎程序，最好是在初始化软盘时，总指定同一个欢迎程序名，这样便于记忆。标准的欢迎程序文件名是 HELLO。

下面我们介绍一下初始化软盘的操作方法。

① 先用 NEW 命令，清除内存中的程序，并把准备初始化的软盘插入到驱动器中，把门关好。

② 将你编写的欢迎程序从键盘上输入到机器内。

③ 为这个程序取一个名字，比如叫“HELLO”，从键盘上打入命令：

INIT HELLO

当你按下回车键以后，会发现软盘驱动器上的红灯马上就亮起来了，随后就听到一阵软盘转动发出的声音，持续几分钟左右的时间，当红灯熄灭，屏幕上再度出现光标时，就完成了初始化工作。

④ 给初始化的软盘准备一张标签，并把盘号及与该盘有关的信息写在标签上。取出软盘，贴上标签，再把软盘放到驱动器中去。

特别要注意不要随便使用 INIT 命令。如果有一张软盘已经初始化，并已存放了重要信息，对这张软盘再进行初始化，将会把原来盘上的所有信息全部清除掉。

所有经过初始化操作的空白软盘都存有 DOS 磁盘操作系统程序，也都可以作为操作系统的引导盘进行启动。现在，我们不要移去新初始化后的软盘，关掉主机电源开关后数秒钟，再重打开主机电源开关，屏幕上将出现如下信息：

THIS IS MY FIRST DISKETTE
INITIALIZED 3/16/88
BY GU XI-XIONG

这表明，我们用初始化后的软盘引导 DOS3.3 操作系统成功。

2.7.4 查看软盘上的文件目录

我们在使用软盘的过程中，往往很希望知道当前驱动器中软盘存放着哪些文件，DOS 的到磁盘文件目录命令可以帮助我们完成这一工作。CATALOG 命令就是用来把软盘上所有文件名的目录送到正在运转的输出设备上，通常是送到屏幕上显示。对于软盘上的每一个文件，CATALOG 命令除列出其文件名外，还列出文件的数据类型，说明文件是否被锁住及文件占用了软盘上的多少扇区等。

现在我们打入命令：

CATALOG

屏幕上将出现如下信息：

DISK VOLUME 254

A 002 HELLO

如果我们换上 DOS3.3 系统主盘，再打入 CATALOG 命令，屏幕上将出现如下信息：

```
DISK  VOLUME  254 ←———①
   *A  008  HELLO
   *I  018  ANIMALS
    T  003  APPLE  PROMS
   *I  006  APPLESOFT
   *I  026  APPLEVISION
   *I  017  BIORHYTHM
   *B  010  BOOT13
   *A  006  BRIAN'S  THEME
   *B  003  CHAIN
   *I  009  COLOR  DEMO
   *A  009  COLOR  DEMOSOFT
   *I  009  COPY
   *B  003  COPY.OBJO
   *A  009  COPYA
   *A  010  EXEC  DEMO
   *B  020  FID
   *B  050  FPBASIC
   *B  050  INTBASIC
   ↑ ↑   ↑       ↑
   ⑤④   ③       ②
```

通过屏幕上所列出的这些信息，我们可以知道以下几件事情：

① 该软盘所占有的卷号：一张软盘只能有一个卷号，它是在初始化命令执行过程中所确定的。

② 文件名：每个文件名使用的字符数应在 1 ～ 30 之间，并且第一个字符必须是英文字母。

③ 文件在软盘上所占用的扇区数：文件占用的扇区数，可用三位数表示。最小的文件（空文件）占用一个扇区，若文件占用的扇区数超过 255，那么，表示的数目将再次由 0 开始计算，但这并不影响此文件占

用扇区的实际位置。软盘上每个扇区可存储多达 256 个字节。

④ 文件类型:文件类型用一个称为类型码的字母来表示。其中:

A — CEC-BASIC 程序文件

I —整数 BASIC 程序文件

B —二进制代码文件

T —文本文件

R —由 EDASM 汇编程序所产生的可浮动目标文件

L —由 LISA 汇编程序所生成的源程序文件

⑤ 文件保护标志:它是操作系统所提供的对文件进行写保护的一项措施。如果该文件有星号标志(*),则表示 DOS 对该文件进行了封锁保护,它只允许用户对该文件执行读操作,而不能对其执行改名,删除或写操作。否则,系统会给出:FILE LOCKED 信息。

在列目录过程中,可以用 Ctrl- S 键暂停列目录,再按任意键使其继续列目录。另外软盘上的程序常常多于屏幕一次能显示的数量,CATALOG 命令会先列出前 18 个文件的目录要看后面其它文件目录时,再按任一键,屏幕将接着显示后面文件的目录。

2.7.5 保存 BASIC 程序到软盘上

当我们在 CEC-I 上编制了一个 BASIC 程序,很希望把它保存到软盘上去,在下次开机时,随后把它调入到主机的内存中来。这时,我们就可以使用 DOS 所提供给我们的 SAVE 命令。其格式为:

SAVE 文件名

该命令将把当前已建立在主机内存中的 BASIC 程序用该文件名所代表的文件保存在软盘上,这个文件名是以字母打头的字符串(但不能含逗号),文件名的最大长度为 30 个字符。

例:我们在主机上打入下面的 BASIC 程序

```
10  INPUT "GUESS A NUMBER:";G
20  READ D
30  IF D=-99999 THEN 80
40  IF G<> D THEN 20
50  PRINT "YOU ARE CORRECT!"
60  END
```

② 将你编写的欢迎程序从键盘上输入到机器内。

③ 为这个程序取一个名字,比如叫"HE

```
80  PRINT"BAD GUESS, TRY AGAIN!": RESTORE:
    GOTO 10
100  DATA 1,393,-39,28,391,0,3.14,90,15,10,5,-34,-99999
```

打入 RUN 命令检查程序执行正确后,可以把该程序保存在工作盘上,我们把该程序取名为:GUESS A NUMBER。可以键入如下命令:

SAVE GUESS A NUMBER

按下回车键以后,驱动器的红灯亮,屏幕上光标消失,几秒钟后,存盘工作完成,光标重新出现,红灯熄灭。

此时,若再键入 CATALOG(列目录)命令,我们将会看到工作盘上已含有两个文件:

```
DISK VOLUME 254
A  002  HELLO
A  002  GUESS A NUMBER
```

利用 SAVE 命令,我们可以在工作盘上保存多个程序文件。换上其它软盘,还可以把这个程序保存到其它软盘上。另外,若所用的文件名在软盘上早已有了,那么,现在存入的程序就会取代原来存入软盘中的同一文件名的程序,老的程序就自动消除。

在使用 SAVE 命令时,我们要注意:保存程序的软盘不能贴有写保护,也不能是没有经过初始化的空白盘片。否则,我们会得到信息:I/O ERROR。

2.7.6 从软盘上读 BASIC 程序

启动 DOS 操作系统后,我们就可以利用 DOS 命令把软盘上的程序装入到主机内存中来。其命令格式为

LOAD 文件名

例如,我们希望把在工作盘上的 GUESS A NUMBER 这个程序装入到主机内存中,则可以键入如下命令:

LOAD GUESS A NUMBER

按回车键以后,CEC-I 就从驱动器中将软盘上的 GUESS A NUMBER 程序载入到主机内存。要证实程序是否已在内

存中，可以在光标重新于屏幕上出现时用 LIST(列程序清单)命令将内存中的程序显示在屏幕上，或者 RUN(运行程序)命令，将读入的程序运行。

在用 LOAD 命令装入程序时，主机内存原有的 BASIC 程序将被清除掉，由新装入的程序替换原有的 BASIC 程序，而软盘上的程序文件不会受任何影响。

在使用 LOAD 命令时，如果软盘目录上没有所指定的文件名，就会得到出错信息，屏幕上显示 FILE NOT FOUND(文件未找到)。如果磁盘上有这个文件名，DOS 就检查该文件的数据形式，如果它不是 CEC-BASIC 文件，则屏幕上将显示出错信息 FILE TYPE MISMATCH(文件类型失配)。

2.7.7 清除软盘上的文件

我们使用软盘来存储信息时，往往希望在盘上保留对我们有用的信息，对于调试程序中留下的不用的旧程序或旧数据文件，我们都可以用 DOS 的删除命令把它们删除，以使软盘上可以留出更多的空间来存放对我们有用的程序或文件。DOS 删除命令的格式为:

DELETE 文件名

例如，我们希望把工作盘上的 GUESS A NUMBER 删除掉，可以键入如下命令:

DELETE GUESS A NUMBER

要注意: 执行 DELETE 命令后，软盘上所指定的文件名将被从文件目录中删除掉，并且不可重新恢复的，所以操作时要十分小心，只有当我们证实了该文件确实对我们无用时，才可将该文件删除掉。在一般情况下为了防止因我们错误的操作而将盘上重要的文件删除掉，应对这些文件用 LOCK 命令加写封锁标志。

为了确认文件是否真的已被删除，可以用 CATALOG 命令检查，如果在目录上再也看不到这些文件。就表明这些文件已被清除掉了。

如果在 DELETE 命令中所指定的文件名不在软盘上，屏幕上将显示: FILE NOT FOUND(文件未找到); 如果软盘上贴有写保护，将显示: WRITE PROTECTED(写保护); 如果 DELELE 命令中所指定的文件名是加了写封锁标志的，将显示: FILE LOCKED(文件被锁

住了)。

2.7.8 更换软盘上的文件名

软盘上的任何文件的文件名都是可以改变的。如果我们在存盘时,把文件名写错了,那么就可以用 DOS 的 RENAME 命令把文件名改正过来。DOS 的更换文件名命令格式为:

RENAME 旧文件名,新文件名

DOS 在执行该命令时不会检查软盘中是否已经有了这个新的文件名。如果盘上已有了这个文件名,那么在命令执行后,就会看到盘上有两个相同的文件名,这就可能产生混乱,而且要想弥补也很困难。

在使用 RENAME 命令时,旧的文件名一定要保证在盘上存在,否则屏幕上会显示出错信息:FILE NOT FOUND。

2.7.9 磁盘文件的加锁与解锁

有时,某个软盘中的程序或资料文件需要永久保存。为此,DOS 提供了一种保护技术,称为文件"加锁"。加锁的文件可以避免删除或者在该文件上又写入新东西等偶然事故的发生。为一个文件的命令格式为:

LOCK 文件名

此后,若对被锁住的文件删除或写入信息,将会得到出错显示 FILE LOCKED(文件被锁住)。

在软盘的目录中,对加锁的文件,在文件类型的前面记有星号(*)的标记。

当需要重写或者删除一个被加锁的文件时可使用 DOS 的解锁命令,把锁消去。解锁命令的格式为:

UNLOCK 文件名

第三章 CEC-BASIC语言程序设计

中华学习机上的 BASIC 语言(CEC- BASIC)固化在主机板上的只读式存储器中。它除了具有 APPLE SOFT BASIC 语言的功能外，还有汉字处理功能，因此可以用 BASIC 语言来处理中文。

在处理中文时，就先按"中文"键，此后就进入中文状态。中文每行显示 17 个汉字或 34 个字符，共有 10 行，第 11 行为状态行。具体汉字的使用操作方法参阅 2.5。在中文状态中，HGR2、FLASH、TEXT、TAB、VTAB、HTAB 这些语句对汉字的显示均有不同程度的影响，下面将作详细的介绍。

另外 CEC- BASIC 增加了音响功能的实现，使用 MUSIC 语句可轻而易举地让计算机奏乐。还增加了 PLAY 语句，增强了 SAVE、LOAD 语句的功能，使 CEC- I 更适合用磁带软件。

3.1 CEC-BASIC 简介

3.1.1 变量与数据的规定

在 CEC- BASIC 中常用一些字符表示变量的名字，所使用的数据都可存放在变量中。变量可分为两类：简单变量与下标变量，下标变量将在 3.6.1 中介绍。使用变量名必须遵守以下规定：

变量名开头一定要是一个英文字母，随后可跟任何英文字母或数字。例：A、BC、D12 等都是变量名。

一个变量 最多可长达 238 个字符。不过计算机只认前两位作为变量名。因此例如 ABCD 与 ABCDEF 实际为同一个变量名。

在变量名中不允许有保留字(保留字见附录 2)，虽然计算机只认变量名的前两位，但整个变量名中不能有保留字出现。例如 FGEND 就不能作为变量名。

变量名分实型、整型与字符型三种，实型变量名中存放的是实数，

整型变量名中存放的是整数，字符串变量名中存放的是由字符组成的串。前面所介绍的为实型变量名。整型变量名在变量名后面加上%，例 AB %。字符串型变量名后面加上 $，例 AB $。

在 CEC- BASIC 中均使用十进制数。实型变量最多可表示九位有效位数的实数，其范围：| 实数| $\leqslant +10^{38}$。BASIC在内部先将十进制数换成二进制数进行计算，算出的结果需输出显示时，再将二进制数转换成十进制数显示出来。由于这些转换带来的误差使之不能得到准确的结果。

例：]PRINT 9 ∧ 2

81.0000001

为了消除这些数制转换运算带来的误差，可用以下公式：

X =INT(X * 10 ∧ D+ 0.5) / INT(10 ∧ D+ 0.5)

其中 D 是小数点后面的位数。

整型变量表示 −32767 ～ + 32767 之间的整数值。整型变量不能用在 FOR 或 DEF 语句中，请用户注意。但在数组运算中使用整型下标变量可节约内存空间。

实型整型数之间可以进行转换，这是由机器自动完成的。实型变量的值负给整型变量时，是取不大于该值的最大整数组整型变量，如同用 INT 取整函数的效果一样。如果整型变量的值负给实型变量，其值没有变化。

例： 10 A =3.65 （变量 A 的值为 3.65）

20 B % =A （把 A 的值赋给整型变量 B % 结果 B % 的值为 3）

字符串变量可表示一个字符组成的串，字符串是用引号括起来的一个字符序列。例："ABCD"。字符串变量也可被赋值，一个串变量可赋 0 ～ 255 个字符。但串变量与数值变量之间不能直接转换。

数值变量在未被赋值前，其值为零；串变量未被赋值前，则为空串。

下面的表把 CEC- BASIC 的三种型态的变量作了小结。

在 CEC- BASIC 中实数的范围在 -10^{38}到10^{38}之间，如果数字的绝对值比 $3*10^{-39}$还小的话，就认为该数为0。整数的范围一定要在 −32767 到 32767 之间才行。实数可有 9 位有效数字。若输入 10 位

有效数字，计算机将对第 10 位进行四舍五入。

描　述	变量后的符号	例　子
实型 (指数在 −38 至 + 38 之间，尾数为 9 位 10 进制)	无	A BOY
整型 (在 −32767 至 + 32767 之间)	%	B % C1 %
串型 (0 至 255 个字符)	$	D $ CITY $

在输出实数时有以下格式限定：

. 如果是个负数，则负号“−”将会打印出来(整数也适用)。

. 如果数字的绝对值是在 0 ～ 999999999 之间的整数，则作为整数完整打印出来。

. 如果数字的绝对值大于或等于 0.01 且小于 99999999.2 则以定点数打印出来，不用指数形式表示。

. 如果数字的范围不在前二项当中，则以科学计数法表示。其格式如下：

S ×.××××××××ESTT

其中 S 分别表示数的符号，指数部分的符号；“+ ”号可以省略；E 表示 10 的幂；× 表示 0 ～ 9 之间的数字；T 表示指数部分的数字。下面几个例子给出了输出格式的形式：

数据	输出格式
+ 1	1
−1	−1
6523	6523
−23.460	−23.46
45.72E5	45.72000
1 * 10 ∧ 20	1E+ 20
−12.34567896 * 10 ∧ 10	−1.23456789E+ 11
1000000000	1E+ 09
999999999	999999999

(注：其中∧表示乘方)

3.1.2 运算符与表达式规则

在 CEC- BASIC 中有三类运算符:

①算术运算符:

+ (加),−(减),∗(乘),/(除),∧(乘方)。

②逻辑运算符:

NOT(非),AND(与),OR(或),=(等于),< > 或 > < (不等于),> (大于),< (小于),> =或= >(大于等于),< =或= <(小于等于)。

③串运算符:

+ (串连接)

用运算符将变量、数值、函数连接起来的有意义的数学表达式称为表达式。因为有三种运算符,因此也就有三种表达式:

串表达式 —— 用串运算符将串变量,字符串数据、串处理函数连接起来的运算式,其中也包括单独的串,串变量。

逻辑表达式 —— 用逻辑运算符将任何种表达式连接起来的运算式。逻辑运算的结果若为真其值为 1;若为假则为 0。

算术表达式 —— 用算术运算符将变量、数值、函数或逻辑表达式连接起来的运算式,其中包括单独的变量、数值或函数。

下面是几个不同类型表达式

A∗3+ (B/2∗SIN(X))	算术表达式
"AB"+ A $ + MID $ (A $,4,2)	串表达式
(A ="YES")AND(B < >9)	逻辑表达式
NOT(5<3)	逻辑表达式
A∗B∗(A > =1)+ A∗(A <1)	算术表达式

各种运算应依照其执行的优先等级由高到低进行运算,对同一等级的运算符的执行是从左到右地执行。优先等级如下:

(,)

+ ,−,NOT 单目运算符

∧

∗,/

+,−

>, <, > =, = >, < =, = <, < >, > <, =

AND

OR

例: 下式的运算顺序为:

```
(10+ 6) * 3 ∧2 * COS(1) / 2 * 8+ 7
    ↓     ↓   ↓   ↓      ↓      ↓   ↓   ↓
    ①    ④、 ③   ⑤      ②      ⑥   ⑦   ⑧
```

3.1.3 保留字

CEC-BASIC 中使用的语句、命令、函数均称为保留字,用户不能使用保留字作为变量名或作为变量名中的一部分。保留字的具体规定见附录 2。

3.1.4 执行方式

CEC-BASIC 有两种执行方式,一种叫做立即执行型,另一种叫暂缓执行型(或叫程序语句型)。

立即型执行方式:不用先键入语句行号,真接键入语句或命令,按下回车键后,计算机立刻执行该语句,随之命令消失,计算机不保存它,但保留其结果,如变量的值。

例:]A =3.14　　　　(把 3.14 赋给变量 A)

]PRINT　A　　　　(打印 A 的值)

3.14

暂缓型执行方式:每个语句前面加上行号,按下回车键后,计算机将此语句存入内存,直到用户键入"RUN"命令时才执行。并可用 LIST 命令查看这些语句行或修改它,这也就是说这些语句是保存在计算机中的。

例:]10　A =3.14

]20　PEINT　A

]RUN

3.14

3.1.5 语句行规则

一个语句行是以逗号开始，直到回车结束。语句行号应是 0 ～ 63999 之间的正整数，一行内最多有 239 个字符，其中包括键入的空格。一个语句行内可有若干个语句，每个语句之间用“:”分隔。

例: 10　A =3.14:PRINT　A

语法定义的有关符号

下面介绍各个语句格式中使用的一些特殊符号，这些符号并不属于 CEC- BASIC 中，但有了它们就能更明确地表示语句的结构或关系。这些符号如下所列:

| 用来分开可任选其中之一的各项。例:a | b 可选择 a，或者选择 b。

〔　〕用来表示括号内的东西是可有可无的。例〔a〕表示 a 可有可无。

{　}用来表示括号内的东西可以重复。例:{a}表示可有一个 a，也可有若干个 a。

3.2 输入与输出语句

利用计算机进行数据处理或运算。首先应把数据提供给计算机，计算机处理后的结果再返回给用户。因此，必须有一类提供数据及输出结果的语句来沟通用户与计算机之间的联系。这类语句在下面分别给予介绍。

3.2.1 LET

执行型:立即与暂缓型

格式:〔LET〕算术变量〔下标〕= 算术表达式

　　〔LET〕串变量〔下标〕= 串表达式

功能:将等号(赋值号)右边表达式的值给左边的变量。LET 可省略。

说明:算术变量未被赋值前其值为 0，串变量为空。

例:
```
]10  PRINT  A,B $
]RUN
0
```

．表达式类型必顺与被赋值的变量类型一致。

例:]A $ =123 ; A B="123"

?TYPE MISMATCH ERROR

这两种都是类型不匹配错误。

．整型变量与实型变量之间赋值时,将自动进行转换与舍取。

．一个变量可被多次赋值,但该变量只保留最近所赋的值。

例:
```
]10  X=2
]20  X=5
]30  PRINT  X
]RUN
5
```

3.2.2 DATA

执行型:暂缓型

格式:DATA 〔数据〕〔{,数据}〕

功能:在计算机内存储数据,以供 READ 语句使用。

说明:程序中可有多个 DATA 语句,且可放在程序的任何位置。该语句中的数据可以是数值,也可以是字符串,但不能是表达式。在字符串中不能出现“,”,若串中必须有“,”号出现,则应在串的两边加双引号,但此时串内不能再出现双引号。否则将引起数据混乱。另外每个 DATA 语句最后一个数据后面不能加“,”。

例:
```
]100   DATA   123, 4.5, ABC"DE, "ABC,DE"
                ↑     ↑      ↑          ↑
              数据1  数据2  数据3      数据4
```

同样也可把这个 DATA 语句拆成多个,其效果是一样的。

```
]100  DATA  123,4.5
]110  DATA  ABC"DE
]120  DATA  "ABC,DE"
```

3.2.3 READ

执行型:立即与暂缓型

格式:READ〔变量名〕〔{,变量名}〕

功能: READ 语句中的变量依次从 DATA 语句中读取数据。

说明: · 变量与读取的数据类型必须一致,否则会出错。

例:
```
]10 READ A,B,C $,D $,E
]20 DATA 4,5,AB,6,E
]RUN
? SYNTAX ERROR IN 10
```

这里 A 变量读取 4,B 变量读取 5,C $ 串变量读取 AB,D $ 串变量读取 6,而 E 变量读取的数据类型与其不符,故出错。

如果想要由 READ 读取比 DATA 更多的数据,则会造成无数据可读的错误。但对多作的数据没有读到则没有关系。

例:
```
]10  READ A,B,C,D
]20  DATA 1,2,3
RUN
?OUT OF DATA ERROR IN 10
```

在立即执行中使用 READ 语句,则必须在暂缓型中有 DATA 语句。

例:
```
]10  DATA  1,2,3
READ A,B,C:PRINT  A ;B ;C
123
```

3.2.4 RESTORE

执行型: 立即与暂缓型

格式: RESTORE

功能: 将数据指针指回数据区的第一个数据位置,使数据可重新使用。

例:
```
]10  READ A,B
20  RESTORE
30  READ C,D
40  PRINT A,B,C,D
50  DATA 7,6,5
RUN
7
```

6

3.2.5 INPUT

执行型:暂缓型

格式:INPUT〔"字符串";〕变量〔{,变量}〕

功能:通过当前输入设备,给变量赋值。

说明:·输入多个数据时,可用逗号分隔每个数据,最后用回车结束。当输入的数据少于变量个数时,屏幕上提示"??",等待继续输入,如果输入的数据个数多于变量个数时,将出现警告性错误:EXTRAIGNOREO,但这不影响程序执行,仅将多余数据忽略。用INPUT语句输入数据时也可用回车键来分隔数据。

例1.10 INPUT A,B,C

RUN

?1,2,3

例2.10 INPUT A,B,C

RUN

?1,2

??3

例3.10 INPUT A,B,C

RUN

?1,2,3,4

EXTRAIGNRED

例4.10 INPUT A,B,C

RUN

?1

??2

??3

·若要中断INPUT语句,应在输入第一个数据之前键入CTRL- C,然后按回车键。

·输入的数据应与变量的类型一致,否则出现"?REINTER",应重新输入数据。

例: 10 INPUT A

RUN

?AB

?REINTER

?

当输入字符串数据时，串内不能有逗号及冒号。若数据内一定要有这两个符号时，则应在串的两边加双引号，但经这样处理后，串内就不能出现双引号。当输入数值数据时只能输入数值，不能输入表达式。

· 若需对所赋值的变量加注释时，可在 INPUT 语句后面的双引号内放入注释内容。

例: 10 INPUT "R=";R

3.2.6 GET

执行型: 暂缓型

格式: GET 变量

功能: 通过当前的输入设备输入一个字符或一个数字赋给对应的变量，但不显示在屏幕上，也无须按回车键。

说明: 如使用算术变量，只能输入一个数字，不能输入","或":"，否则会出现 EXTRA IGNORED；如果输入＋、−、.、E 那么实际赋给变量的值为 0；如果输入除上述字符以外的其他非数字字符时，将显示出句法错误，基于上述原因，最好不要使用算术变量。由于 GET 语句对输入的字符不显示，因此在程序设计时常与 PRINT 语句连用。

例: 10 GET A $: PRINT A $

3.2.7 PRINT

执行型: 立即与暂缓型

格式: PRINT 〔{表达式}〔,|;〔{表达式}〕}〕〕〔,|;〕

功能: 将表达式的值或数据输出到所指定的输出设备上。

说明: 表达式包括数学表达式和字符串表达式。表达式之间可用分隔符(,或;)隔开，这将影响输出的格式，若用分号作分隔符，则前后表达式的结果是紧接着输出的；若用逗号作分隔符则按标准格式输出。每一行分为三个区，第一区占 16 个字符位置(位置 1 到 16)，第二区占

16 个字符位置(17 到 32),第三区占 8 个字符位置(32 到 40)。若在第三区所输出的字符个数超过 7 个时,则第三区不占用,而移到下一行第一区输出这些字符。

例 1、10　PRINT　5+ 8 ;A ;“A”

RUN

130A

例 2、10　PRINT　5+ 8, A, “A”

RUN

13	0	A
↑	↑	↑
第 1 列(位置 1)	第 17 列(位置 2)	第 33 列(位置 3)

例 3　10　PRINT　5+ 8, A, 3.1415926, “A”

RUN

13　　　　　　　0

3.1415926　　　A

其中第三个输出内容超过 7 个字符,故在下一行第 1 个位置输出。

可用?作为 PRINT 的缩写。

PRINT 语句末端标点符号的使用:当 PRINT 语句末端没有标点符号时,执行下条 PRINT 语句将另起一行输出。

例:　10　PRINT “ABC”

20　PRINT “DEF”

RUN

ABC

DEF

当前一条 PRINT 语句的末端以“,”结尾时,下一条 PRINT 语句所输出的内容要紧接着前一条输出的内容,中间不留间隙。

例: 10　PRINT “ABC”;

20　PRINT “DEF”

RUN

ABCDEF

当前一条 PRINT 语句的末端以“,”结尾时,输出完本句的数据

后，下一条输出语句将从该行的下一个标准位置继续输出。

例: 10 PRINT "ABC",

20 PRINT "DEF"

RUN

ABC   DEF

3.2.8 IN#

执行型：立即与暂缓型

格式：IN# 表达式

功能：选择输入设备。

说明：根据表达式的值选择输入通道号。其值在 0 ~ 7 之间。CEC 机规定当值为 6 时启动磁盘驱动器进行读盘操作，当值为 3 时，进入汉字状态；当值为 0 时，以键盘作为输入设备。进入汉字系统后，不能使用此命令。

例: IN#3 随之入汉字系统。

IN#3

3.2.9 PR#

执行型：立即与暂缓型

格式：PR# 表达式

功能：选择输出设备。

说明：根据表达式的值选择输出通道。当其值为 0 时，以 CRT 或电视机作为输出设备。其它与 IN 相同。

例: PR#6

表示启动磁盘驱动器。

3·3 和顺序有关的语句

下面所介绍的语句，在使用它们时可以改变程序的执行顺序，从而使程序产生分支，循环等。

3.3.1 GOTO

执行型:立即与暂缓型

格式:GOTO 行号

功能:执行该语句时,计算机自动转向行号所指定的语句上去执行。

说明:如果该行号无语句时,将产生出错信息。

例:GOTO 5000

3.3.2 IF......THEN

执行型:立即与暂缓型

格式:IF 表达式 THEN 语句〔{:语句}〕

IF 表达式 THEN 行号

IF 表达式 GOTO 行号

功能:根据表达式的值确定程序的转移方向。

说明:如果表达式的值不为 0,则就认为表达式为真,继而执行 THEN 后面的指令。否则将忽略 THEN 后面的任何指令,而执行下一程序行。

决定表达式值是否为零的方法如下:

· 算术表达式的绝对值小于等于 2.93873E−39 时,其值为 0。

· 串表达式为空时,其值为 0。

· 逻辑表达式的关系不满足时,其值为 0。

例 1. 100 IF A+ B THEN 20

若 A+ B 的值大于 2.93873E−39 时就转向 20 程序行。

例 2. 100 IF A $ THEN PRINT A $

若 A $ 不为空串时,则打印 A $ 的值。

例 3. 100 IF X <5 THEN GOTO 500

若 X <5 (条件为真),则执行 500 程序行。

注意在 THEN 之前不能使用字母 A。

3.3.3 FOR、NEXT

执行型:立即与暂缓型

格式:FOR 实型算术变量 = 算术表达式 1 TO 算术表达式 2

〔STEP 算术表达式 3〕

⋮
循环体
⋮

NEXT〔算术变量〕〔{,算术变量}〕

功能:循环执行 FOR 与 NEXT 之间的程序行。

说明:· 算术表达式 1 称为循环的初值,算术表达式 2 称为循环的终值、算术表达式 3 称为循环步长,实型算术变量称为循环控制变量。循环时先将初值赋给循环控制变量,同时计算机记录循环的终值与步长,然后执行循环体内容,执行到循环尾 NEXT 语句时,把循环控制变量的值加上步长再赋给循环控制变量再判断。若该变量的值满足下列关系式:SGN (步长) * (变量的值)>ABC(步长)时就退出循环,否则就继续执行循环体的内容。其中 ABC 是绝对值函数,SGN 是符号函数,这两个函数将在 3.9.1 介绍。

· 执行 FOR、NEXT 语句时应满足:初值×步长< =终值×步长时,才能进行正常的循环。

例:

```
10  FOR  I=1  TO  10  STEP  -1
20  PRINT  "* * *"
30  NEXT  I
40  PRINT  "$ $ $"
  RUN
  * * *
  $ $ $
```

此程序就不能正常的循环。

· 循环语句中的初值、终值和步长可以是变量或表达式。

例:

```
10  A=1 : B=5
20   FOR I=A  TO  B STEP  A+ 1
30  PRINT  I,
40  NEXT  I
50  END
RUN
1                  3                  5
7
```

· 循环语句中的终值与步长是在计算机执行 FOR 语句时记录下来的，循环体中的任何语句均不会对它产生任何影响。

例:
```
10  A=1 : B=5
20  FOR I=A  TO  B  STEP  A+ 3
30   PRINT  I
40  A=10 : B=100
50   NEXT  I
RUN
1                5
```

· 循环体内的语句可改变循环控制变量的取值，从而影响循环次数。

例:
```
10  FOR  I=1  TO  10 * I  STEP  1
20   PRINT  I,
30   I  =2 * I
40   NEXT  I
RUN
1               3               7
```

· 步长为 1 时，STEP 1 可省略不写。

例:
```
10  FOR  I=1  TO  10 * I
```

· 循环可以嵌套，但最多不能超过 10 层。

例:
```
10   FOR  I=1  TO  10            10   FOR  I=1   TO  10
20    FOR  J=1  TO   10          20    FOR  J=1  TO  10
30     FOR  K=1  TO  10    或    30     FOR   K=1  TO  10
             ⋮                                ⋮
400    NEXT  K                   400   NEXT  K,J,I
410   NEXT  J
420  NEXT  I
```

· 多重循环不允许分支。在程序中允许从循环体内转到循环体外，而不允许从循环体外转移到循环体内。

例:

```
10  FOR I=1 TO 10              10  FOR I=1 TO 10
         .                              .
         .                              .
         .                              .
50  FOR J=1 TO 10              50 ......
         .                              .
         .                              .
         .                              .
100  IF .....THEN 250    或    100  FOR J=1 TO 10
         .                              .
         .                              .
         .                              .
200  NEXT J                    200  IF THEN 50
250  .....                              :
300  NEXT I                    300  NEXT :NEXT
```

上面程序是允许的。

```
10  FOR  I=1 TO 10             10  FOR I=1 TO 10
20  FOR  J=1 TO  10            20  GOTO 50
         .               或    30  FOR J=1 TO 10
         .                              .
         .                              .
40  NEXT I                              .
50  NEXT J                     400  NEXT : NEXT
```

上面程序是不允许的。

· 在立即执行方式中 FOR 与 NEXT 语句必须放在同一行中，且一行最多不能超过 239 个字符。

· 注意在 TO 的前面不允许有 A 字符。

3.3.4 GOSUB RETURN

执行型：立即与暂缓型

格式：GOSUB 行号

```
        .
        .
        .
```

RETURN

功能：执行 GOSUB 语句时，转到行号为首行的 BASIC 行程序去执行，直到遇到 RETURN 语句，又返回 GOSUB 下一个语句行继续执行。

说明：每次执行 GOSUB 时将该语句之后的程序行地址压入堆栈，然后执行子程序。在子程序中遇到 RETURN 或 POP 时，再将堆栈中保存的返回地址弹出，使程序跳到最近执行的 GOSUB 语句的下一程序行继续执行。

GOSUB 中的行号必须在程序中存在。

子程序嵌套最多不可超过 25 层。

如果未执行 GOSUB 语句而直接执行 RETURN，则出现 RETURN WITHOUT GOSUB ERROR。

基于这两个语句的特点，程序中有不同地方要做同样的工作时，使用 GOSUB 与 RETURN 语句表构造子程序是最适宜不过的了。

例：计算 6! + 9! / 5! 时，可将求 N! 作为子程序进行处理。下面的程序中 100 ～ 140 程序行为子程序，在主程序中调用了三次子程序。

```
10 A = 6:B = 9:C = 5:D = 0
20 N = A: GOSUB 100:D = D + S
30 N = B: GOSUB 100:D = D + S
40 N = C: GOSUB 100:D = D / S
50  PRINT "(6!+9!)/5!=":D
60  END
100 S = 1
110  FOR I = 1 TO N
120 S = S * I
130  NEXT I
140  RETURN

]RUN
(6!+9!)/5!=3030
```

3.3.5 POP

执行型：立即与暂缓型

格式: POP

功能: 将堆栈中的地址弹出，但不返回最近执行的 GOSUB 的下一个程序行，而直接执行 POP 的下一语句行。

说明: 如果未执行 GOSUB 而直接执行 POP 也将出现 RETURN WITHOUT GOSUB ERROR。

例: 下面的程序为求任意阶乘的程序，在 100 程序行中对 N 进行判断，当 N <0 或 N >33 (当 N =34 时，34! 的值太大，计算机已上溢出)时使用 POP，并直接转入第 10 程序行执行。

```
10   INPUT "N=":N
20   GOSUB 100
30   PRINT N;"!=";S
40   GOTO 10
100   IF N < 0 OR N > 33 THEN  POP
       : GOTO 10
110 S = 1
120  FOR I = 1 TO N
130 S = S * I
140  NEXT I
150  RETURN
]RUN
N=5
5!=120
N=-5
N=6523623
N=8
8!=40320
```

3.3.6 多向转移和多向转子语句

执行型: 暂缓型

格式: ON 表达式 GOTO 行号 1, 行号 2, ……行号 i

ON 表达式 GOSUB 行号 1, 行号 2, ……行号 i

功能: 根据表达式的值 n，选择第 n 个行号，使计算机转向该行号去执行。

说明: 算术表达式的值应在 0 ~ 255 之间，并具有自动取整功能，否则将出现?ILLEGAL QUANTITY ERROR 。如果算术表达式的

值为 0,或超过行号组的行号个数时,就忽略此语句,而继续往下执行。这常在菜单选择程序中使用。

例:
```
10  INPUT "PLEASE CHOOSE(1-5):";A
20  IF A <0 OR A >5 THEN 10
30  ON A GOTO 1000,2000,3000,4000,5000
         .
         .
         .
```

计算机根据用户的选择而转向不同的程序段执行。

3.3.7 ONERR GOTO

执行型:暂缓型

格式:ONERR GOTO 行号

功能:程序出错后转出错处理程序。

说明:当程序运行过程中,产生错误时,该语句可防止打印出错信息与中断程序,并自动转到行号所指定的错误处理程序执行,同时它也可防止 CTRL-C 中断程序运行,这对 BASIC 程序的保密有一定作用,发生错误时,错误类型码存放在第 222 号单元($ DE)中,下表列出各类错误的编码表,用户可用 PRINT PEEK(222)得到错误类型码。

类型码	出错情况
0	NEXT 与 FOR 不匹配
16	句法错
22	RETURN 与 GUSUB 不匹配
42	缺少数据
53	非法量
69	溢出
77	缺少内存
92	未定义语句
107	不适当下标
120	重定义数组
133	除数为零
163	类型不匹配

176　　　　串太长
191　　　　形式太复杂
224　　　　未定义函数
254　　　　响应 INPUT 语句时输入不适当
255　　　　用 CTRL－ C 中断纠错程序

例：下面这个程序仍然是求 N! 的程序。在此程序中增加了一个出错处理语句— 0 程序行。当输入的数太大，产生上溢出时，就转向 1000 程序行，显示出 N 太大了。

```
]LIST

0  ONERR  GOTO 1000
100  INPUT "N=":N
110 S = 1
120  FOR I = 1 TO N
130 S = S * I
140  NEXT I

160  PRINT N;"!=";S
170  END
1000  PRINT "IT'S TOO LARGE!!"
]RUN
N=4545
IT'S TOO LARGE!!
```

3.3.8 RESUME

执行型：暂缓型

格式：RESUME

功能：返回当初发生错误处继续执行。

说明：此语句应放在错误处理程序的最后，若未发生错误而执行该语句时则会造成?SYNTAX ERROR IN 65278 使发生错误。若在错误处理程序中也产生错误，执行 RESUME 将会使程序发生死循环。

例：这是一个求平方根程序，当输入的数为负数时，执行 110 程序行出错后，转入 1000 程序行，显示出 N 是负数，并重新输入一个数，然后计算机再返回 110 程序行执行。

```
0  ONERR  GOTO 1000
100  INPUT "N=":N
110 S =  SQR (N)
120  PRINT "SQR(";N;")=";S
130  END
1000  PRINT "N IS NEGATIVE !!"
1005  INPUT "N=":N
1010  RETURN
```

3.4　和系统有关的语句

3.4.1 LOAD 与 SAVE

执行型:立即与暂缓型

格式:LOAD〔"文件名"〕

SAVE〔"文件名"〕

功能:LOAD 是将磁带上的 BASIC 程序装入内存。

SAVE 是将内存中的 BASIC 程序存入磁带。

说明:程序存入磁带的过程是先键入 SAVE 和文件名,同时将磁带机的 PLAY 与 REC 键按下,然后按回车键。在听到两个"嘟"声后,表示装带完毕,同时显示 BASIC 提示符,这时再按下磁带机的 STOP 键。

将磁带上的程序装入内存的过程是先键入 LOAD 与文件名,同时按下磁带机的 PLAY 键,然后按回车键。在这之后机器将发出一声"嘟",装完后显示出装入的文件名及 BASIC 提示符,当装入的文件与所需的文件名不相符时,屏幕上显示出"HAS BEEN LOADED",并继续装入下一个程序......直到找到文件为止。

这两个命令的后面也可不跟文件名,这时 LOAD 只将当前第一个文件调入内存。

注意:在用 LOAD 调文件装入内存时,由于各种类型的磁带机(可以是普通的录音机)的电性能的差异必须把磁带机的音量开关调整到合适的位置,以保证合适的输入、输出电平。在初次使用磁带机时应多试

验几次，当装带成功时，应把音量旋钮开关的位置记录下来。

3.4.2 NEW

执行型：立即与暂缓型

格式：NEW

功能：删除当前内存中的 BASIC 程序及变量。

说明：在输入新 BASIC 程序之前必顺使用 NEW 命令，清除内存中原有的程序。这样才能保证新的程序正常运行。

3.4.3 RUN

执行型：立即与暂缓型

格式：RUN 行号

功能：运行一个 BASIC 程序。

说明：执行 RUN 命令后，请除所有变量、指针、堆栈、并从指定的行号开始执行。如没有指定行号，那么就从程序的最小行号开始执行。

3.4.4 STOP

执行型：立即与暂缓型

格式：STOP

功能：中止程序执行。

说明：执行此命令后，显示出“BREANK IN 行号”，并把系统控制权交还给用户。利用 STOP 语句，用户可以检查和修改程序。当程序需要执行时，可用 GOTO 或 CONT 命令，这时程序可接着运行，而 STOP 之前各变量的值并不改变。

例：

```
10  INPUT "A, B="; A, B
20  C=A+ B : PRINT C
30  STOP
40   PRINT SQR (C)
50   GOTO 10
RUN
A, B=10, 6
```

```
16
CONT
4
A,B=-10,6
-4
GOTO 50
A,B=
      ·
      ·
      ·
```

程序在 STOP 之前打印出 C 的值,STOP 执行后,由用户自行确定程序的走向。

3.4.5 END

执行型:立即与暂缓型

格式:END

功能:终止程序执行。

说明:一般 BASIC 程序也可以没有 END 语句作为程序的结束,但在具有子程序的程序中没有 END 语句,有可能直接运行子程序而出错。

3.4.6 CTRL-C

执行型:立即型

功能:中断程序执行。对 INPUT 语句,如果输入的第一个字符是 CTRL- C,并按回车键,就中断 INPUT 语句的执行。

说明:按下 CTRL 键,同时按下 C 键。这在调试程序,特别当一个程序进入死循环时常使用它。

3.4.7 CONT

执行型:立即与暂缓型

格式:CONT

功能:如果程序是由 STOP, END 和 CTRL- C 停止运行的,那

么 CONT 命令可使程序从下一语句开始继续执行。

说明:如果 INPUT 语句被 CTRL- C 中断的话,再用 CONT 命令会造成“SYNTAX ERROR IN 行号”。如果键入 CONT 之后,机器显示出”?CAN'T CONTINUE ERROR”那么表示程序在中止执行后,用户可能有以下操作。

改变或删除了程序中的某几行;作了造成错误的动作。

例题见 STOP 语句中所举的例子。

3.4.8 CTRL-RESET

执行型:立即型

功能:立即中止程序或命令执行。

说明:将 CTRL 与 RESET 键同时按下,执行此命令可将有些指针及堆栈表清除,因此不能用 CONT 继续执行,但程序仍在内存中,未丢失。

3.4.9 TRACE 与 NOTRACE

执行型:立即与暂缓型

格式:TRACE

NOTRACE

功能:TRACE 是跟踪程序的执行,并将执行过程中的行号显示在屏幕上;NOTRACE 是解除跟踪。

说明:这是一对很有用的命令,利用它可帮助调试程序。

例:

```
10  FOR I = 1 TO 10
20  PRINT I
30  NEXT I
40  END

]RUN
#10 #20 1
#30 #20 2
#30 #20 3
#30 #20 4
#30 #20 5
#30 #20 6
```

```
#30 #20 7
#30 #20 8
#30 #20 9
#30 #20 10
#30 #40
```

3.4.10 POKE

执行型:立即与暂缓型

格式:POKE 算术表达式 1,算术表达式 2

功能:把算术表达式 2 的值写入以算术表达式 1 为地址的内存单元。

说明:两个表达式均是十进制数,其中表达式 2 的值必须在 0 ~ 255 之间,表达式 1 的值必须在 −65535 ~ 65535 之间。利用 POKE 语句可建立机器语言程序,并可修改某些内存单元的值,但在内存中有很多地址内存放一些极重要的信息,使得机器正常工作。若用 POKE 命令把这些地址的内容改变后,将会影响整个系统的正常工作。因此使用 POKE 时应特别谨慎。

例:把一 BASIC 程序键入内存,并调试好后,再键入 POKE 214,128,此后对键入的任何其他命令,机器都视为"RUN"命令,执行程序,从而起保密作用。

3.4.11 WAIT

执行型:立即与暂缓型

格式:WAIT 表达式 1,表达式 2〔,表达式 3〕

功能:此命令允许用户在程序中插入有条件性的暂停。

说明:表达式 1为内存中的一个地址,其范围在 −65535 ~ 65535 之间。表达式 2 与表达式 3 都是十进制正数,从 0 ~ 255 之间,在执行 WAIT 语句时应将这两个数转换成二进制数,即 0 到 11111111。如果该语句中只使用两个参数,即表达式 1 与表达式 2,那么执行该语句时将表达式 1 的地址中的值与表达式 2 的值相与(二进制数每一位相与,也就是当两个二进制数对应的字符都为 1,则相与后该位为 1;只要有一个对应字符为 0,则相与后该位为 0。例:若表达式 1 的地址中的值为 10111101,表达式 2 的值为 10011111,这两个数相与后的值为 10011101,若结果 8 位二进制全为零则重复这个过程;若结果为非零,程序便向下继续执行)。在实际使用中表达式 1 地

址内的值是在变化的，当满足某个条件时，使机器暂停。

例: WAIT 算术表达式 1,3

只在算术表达式 1 地址的值最右边两位中至少有一个 1，程序才能继续往下运行，否则程序停顿下来等待。

而 WAIT 算术表达式 1,0

将使机器永远停顿下来，因为算术表达式 1 地址的值与 0 相与后，其结果衡为 0。

如果该语句有三个参数，那么先将算术表达式 1 的地址内容与算术表达式 3 的值作异或(即每一位相异或。所谓异或是：算术达式 1 地址中的值与算术表达式 3 的值化成二进制数后，如果它们对应的位一样，那么该位的结果为 0，如果它们对应位不一样，那么结果为 1。例：10011010 与 11010110 相异或后的值为 01001100)，接下来，把异或后的结果再与表达式 2 的值相与。若相与后的结果为零，则重复做这种测试过程；若结果为非零就继续往下执行。

以下举几个例子，使 WAIT 更容易理解。

WAIT 算术表达式 1,255,0

只有当算术表达式 1 地址的二进制数值中，有一位或更多位为 1 时，程序就继续执行，否则程序暂停。

WAIT 算术表达式 1,255

这个语句与上一个效果一样，因为机器自动认为算术表达式 3 的值为 0。

WAIT 算术表达式 1,1,1

只有当算术表达式 1 地址的二进制值中至少有一位 0 时，程序才继续往下执行，否则程序将暂停。

WAIT 算术表达式 1,1,1

只有当算术表达式 1 地址的二进制值中最右边的位为 0 时，程序才继续往下执行，否则程序暂停。

WAIT 算术表达式 1,3,2

只有当算术表达式 1 地址的二进值中最右边的位为 1，或最右边第二位为 0，或这两种情况都满足时，程序才继续往下执行，否则程序暂停。

下面的程序段中，只有键入一个字符后，程序才继续往下执行，否

则机器一直等待。

```
10   POKE -16386,0
20  WAIT -16384,128
         ⋮
```

3.4.12 CALL

执行型: 立即与暂缓

格式: CALL 算术表达式

功能: 调用以算术表达式的值为入口地址的机器语言。

说明: 由于有了这个命令,使 BASIC 程序中可调用机器语言子程序。算术表达式的值范围在 -65535 ~ 65535 之间。在被调用的机器语言子程序最后应有 RTS 返回指令才能返回到 BASIC 程序,并继续执行下面的程序。

例: CALL-936

调用入口地址为 -936 的机器语言子程序,执行的结果,如同 HOME 命令一样。

3.4.13 HIMEM

执行型: 立即与暂缓型

格式: HIMEM: 算术表达式

功能: 用来设置 BASIC 程序运行时内存中可用到的最高地址。

说明: 算术表达式的值应大于数组尾地址,否则程序无法运行。且该值不得大于内存初始化时的 HIMEM 值。在引导 DOS 系统后, HIMEM 值自动设置为 $ 9600; 在不装入 DOS 情况下, HIMEM 值为 $ C000; 在 CEC- BASIC 中文状态下由于 $ 9200 到 $ 9600 为中文屏幕映射区,所以进入中文状态后机器自动设置为 $ 9200。用户在设计 BASIC 程序时,可根据程序的结构,调整 HIMEM 值以保护高分辨率图形区,也可将机器语言子程序放置在 HIMEM 设置的地址之上的空闲区中,使之免遭破坏。使用时一般将 HIMEM 放在程序的最前面,至少应放在 BASIC 字符串出现之前。HIMEM 的当前值存放在

116（$ 74）与 115（$ 73）单元中，用户可键入 PRINT PEEK（116）＊256+ PEEK（115）得到当前值。

3.4.14 LOMEM

执行型：立却与暂缓型

格式：LOMEM：算术表达式

功能：用以给出变量表的首地址，用来设置 BASIC 程序运行时内存可用到的最低地址。

说明：算术表达式的值应大于程序的尾地址且小于 HIMEM 的值。这个命令常用于保护高分辨图形。因为一个长程序或变量较多的话运行时往往会把变量表或数组表延伸到高分辨率图形区，这样使图形或程序遭到破坏，用这个命令可命名这些表设置到图形区之后，保护图形区。例：LOMEM：24576 可使变量表设置到 $ 6000 之后。使用该命令时一般将 LOMEM 放在程序之首。若用户未设置 LOMEM 值，程序的尾地址加 2 即为 LOMEM 值。另外 LOMEM 值随程序的增加、删减自动调整。LOMEM 的前值存放于 106（$ 6A）与 105（$ 69）单元中，用户可用 PRINT PEEK（106）＊256+ PEEK（105）得到当前值。

3.4.15 PLAY

执行型：立即型

格式：PLAY

功能：将盒式磁带机中的软件装入内存并自动运行。

说明：这个命令适用于专门的软件公司或计算机厂为中华学习机生产的磁带软件。使用这种软件的操作方法：先将磁带退到开始位置，在键盘上键入 PLAY 四个字符，按下磁带机的“PLAY”键，同时按下键盘上的回车键（RETURN）。此时屏幕显示出“WAITING...”信息。当磁带中的程序装入机器后自动运行。最后按下磁带机的“STOP”键。此命令的实质是将磁带中的程序读到内存的$ 300 ～$ 3FF 中，由于第 0 页与第 2 页中的某些单元的作用，使以后装入内存的程序能够自动执行。

3.5 编辑与显示格式语句

3.5.1 LIST

执行型：立即与暂缓型

格式：LIST〔行号 1〕〔行号 2〕

　　　LIST〔行号 1〕〔，行号 2〕

功能：列出当前内存中的 BASIC 程序清单。

说明：将大于等于行号 1 到小于等于行号2 的程序清单列出。LIST 后边无行号或行号为 0 则将全部清单都列出。

3.5.2 DEL

执行型：立即与暂缓型

格式：DEL 行号 1，行号 2

功能：删除行号 1 到行号 2 之间的 BASIC 程序行。

说明：在暂缓执行型中使用 DEL 命令时，应尤为慎重，否则将中断程序执行，甚至破坏程序。

例：源程序为

```
10  FOR I=1 TO 10
20   A=A+I
30  S=S*A
40  PRINT  S
50  NEXT I
]DEL 20,30
]LIST
10  FOR I=1 TO 10
40   PRINT S
50  NEXT I
```

3.5.3 REM

执行型：立即与暂缓型

格式：REM〔字符串〕

功能：注释一个程序段，使用户了解程序段的作用。

说明：REM 后面可跟任意字符，包括，与：等，最后以回车键结束。因此 REM 之后不能再有其它语句。

例：
```
10  REM THIS PROGRAM IS N!
```

```
20  INPUT "N=";N
30  S=1
40  FOR I=1 TO N
50  S=S*I
60  NEXT
70  PRINT N;"!=";S
80  END
```

10 程序行注释下面的这个程序段是求 N! 的程序。

3.5.4 VTAB

执行型: 立即与暂缓型

格式: VTAB 算术表达式

功能: 把屏幕光标移到算术表达式的值所指定的行上。

说明: 在西文状态下,表达式的值在 1 ~ 24 范围内; 在文状态下,该值应在 1 ~ 10 范围内。并具有自动取整的功能。

3.5.5 HTAB

执行型: 立即与暂缓型

格式: HTAB 算术表达式

功能: 把屏幕光标移到算术表达式的值新指定的列上。

说明: 算术表达式的值应在 0 ~ 255 之间,在西文状态下光标所在行的 1 ~ 40 列为前行,41 ~ 80 列为下一行; 中文状态下光标所在行的 1 ~ 34 列为当前行,35 ~ 68 列为下一行; 当该值为 0 时则将光标移到本行的第 256 位置上。并具有自动取整的功能。

例: 下面的程序运行后可打印出 SIN 函数曲线。

```
5  HOME
10  FOR X = 0 TO 39
20 Y =  -  SIN (X / 20 * 3.14) *
     10 + 12
30  VTAB Y: HTAB X + 1
40  PRINT "*"
50  NEXT X
```

3.5.6 HOME

执行型：立即与暂缓型

格式：HOME

功能：清除屏幕，光标移到屏幕的左上角。

说明：这个命令没有清内存的作用，程序虽然在屏幕上消失，而仍在内存中。

3.5.7 CLEAR

执行型：立即与暂缓型

格式：CLEAR

功能：清除任何变量及字符串等，并将各种指针，堆栈归零，但程序仍在内存中。

例：
```
]A =450: PRINT  A
450
]CLEAR: PRINT  A
0
```

3.5.8 FLASH

执行型：立即与暂缓型

格式：FLASH

功能:执行此命令,会使以后显示在屏幕上的字符闪烁。

说明:中文状态下不执行此命令。

3.5.9 INVERSE

执行型:立即与缓型

格式:INVERSE

功能:使以后显示在屏幕上的字符以反相显示出,即“白底黑字”。

3.5.10 NORMAL

执行型:立即与暂缓型

格式:NORMAL

功能:使屏幕字符以正常方式显示,即“黑底白字”。

说明:在使用 FLASH 及 INVERSE 命令后,若需正常显示字符,就用此命令以恢复正常显示。

3.5.11 SPEED

执行型:立即与暂缓型

格式:SPEED=算术表达式

功能:设置屏幕送出字符的速度,或其他输入、输出设备的速度。

说明:表达式的值范围在 0 ~ 255 之间,最低速为 0,最高速为 255。机器在用户未设定值时为 255。

例:运行下面的程序后,可看到显示字符的速度随 SPEED 的值而变化。

```
10  FOR S=0 TO 255 STEP 5
20  SPEED=S
30  PRINT "ZHONG HUA XUE XI JI"
40  NEXT S
```

3.6 定义语句

3.6.1 DIM

执行型:立即与暂缓型

格式:DIM 变量名(下标)〔{,变量名(下标)}〕

功能:定义一个数组,并给数组在内存中开辟一个数据区。数组中的每个元素作为变量使用。

说明:数组中的元素也称为下标变量,因此下标变量也是变量的一种表示形式。如 A(7)就是一个下标变量。对下标变量有如下规定:

在括号左边的称为下标变量名,下标变量名必须是由英文大写字母开头,后边可跟字母或数字,与简单变量名的规定相同。

在括号内称为下标,下标只能放在括号内。如:

A(I),B(7),C(I,J),D1(3,7)

· 下标可以是数值,变量或表达式。如:

A(8),M(I),B(I * J,J)

· 下标还可以是下标变量.如:

A(B(1),B(2)),C(D(1) * D(2))

· 下标只能是正实数或零,不能是负数。若下标中带有小数,机器具有自动取整动能。如:A(1.5)实质为 A(1)。

下标变量在使用之前,必须用 DIM 语句对所使用的数组进行定义。数组可以是一维,也可以是多维,视用户的需要而定,但数组维数最多不能超过 88 维,即 DIM A(n1,n2......n88),在数组中可使用元素的多少是由可用内存单元来决定的,它取决于 LIMEM,LOMEM 及程序中使用简单变量的多少。数组中每一维最小下标是零。例如 DIM A(2)实际有 3 个元素,即 A(0),A(1),A(2)。每一维的下标值最大可达 32767,即可定义 DIM A(32767)。数组与简单变量一样也有三种类型:整型、实型与字符串型。

例: DIM A %(S),B $ (5),C(S)。

在使用数组元素之前应先定义数组名称,但如果使用的元素下标值均在 0 ~ 10 之间,则可不定义数组,直接使用。一个数组只能定义一次,否则将出现 REDIM ARRAY 错误。

数组最适用于数据的处理。

例:随机产生 N 个随机数,然后按这些数的大小顺序排列打印出来

```
]LIST

10   INPUT N
20   DIM A(N)
```

```
30  FOR I = 1 TO N
40 A(I) =  RND (1)
50  NEXT I
60  FOR I = 1 TO N - 1
70  FOR J = I + 1 TO N
80  IF A(I) > A(J) THEN 100
90 C = A(I):A(I) = A(J):A(J) = C
100  NEXT J,I
110  FOR I = 1 TO N
120  PRINT A(I)
130  NEXT I
140  END

]RUN
?10
.936183085
.908250485
.791714648
.779829242
.673272226
.413178555
.405076424
.354899917
.293299792
.0768111052
```

3.6.2 DEF FN

执行型:暂缓型

格式:DEF FN 变量名(实型算术变量)=算术表达式

功能:用户可自定义一个函数,以后在程序中可直接使作这个函数。

说明:先用 DFE 定义函数 FN 的名称,格式中的实型变量代表自

变量,FN 变量名表示自变量的函数,它们的函数关系由等号右边表达式给出。一旦函数定义后,FN 变量名(自变量)的函数应可使用,取不同自变量的值就可得到一一对应的函数值。

例:
```
10  DEF FNP (X) =1 / SIN(X)
20  FOR I=1 TO 2
30  INPUT X
40  PRINT X,FN P(X)
50  NEXT I
]RUN
?1.507
1.507              1.00203844
?3.14
3.14               627.883015
```

使用自定义函数应注意:

· 只能定义算术函数,不能定义字符串函数。

· 使用自定义函数之前必须先定义函数。

· 自定义函数是一个一元函数,即只有一个自变量,但在函数的表达式中自变量出现的次数无限制.例 DEF FNY(X) =X ∧ 3+ 3 * X ∧ 2+ 3 * X+ 1。

· DEF 语句中的自变量是一个虚变量,在调用函数时用实变量或其他参数值代替。但自变量一定为实型数。

· 对同一个自定义函数可多次重新定义。

3.7 文本、图形与音响语句

3.7.1 TEXT

执行型:立即与暂缓型

格式:TEXT

功能:学习机中有三种不同的显示方式:文本显示方式、低分辩率图形显示方式及高分辩率图形显示方式。执行该语句使低分辩率图形方式或高分辩率图形方式转换到文本显示方式(每行 40 列,共 24 行)。

说明:在西文文本方式中,执行 TEXT 和 VTAB24 命令的效果相

同。西文文本显示区的内存地址为＄ 400 ~ ＄ 7FF；中文文本显示区的内存地址为＄ 4000 ~ ＄ 5FFF。

3.7.2 GR

执行型：立即与暂缓型

格式：GR

功能：把屏幕设置为低分辨率图形方式(40 × 40)并在屏幕下方留 4 行作为文本窗口。

说明：执行此命令后，屏幕清除为黑色，原低分辨率图形区的图形被破坏，光标移至文本窗口内。此时 COLOR 被设置为 0(黑色)，因此图形颜色只能在 GR 语句之后设置。

要使图形、文本混合方式转到纯低分辨率图形方式，可用 POKE 或 PEEK 命令访问地址(-16302)或等效地址(49324)，如：POKE-16302，0。再返回混合方式可访问地址(-16302)或等效地址(49325)。

在程序中要由高分辨率第二页 HGR2 转到 GR 时，应先使用 TEXT 命令。

低分辨率图形显示区的内存地址为＄ 400 ~ ＄ 7FF，从这里可看出文本与低分辨率图形区是在同一区域。

3.7.3 COLOR

执行型：立即与暂缓型

格式：COLOR = 算术表达式

功能：在低分辨率图形方式中，用 COLOR 来设置颜色进行绘图。

说明：机器对算术表达式的值自动取整，其值应在 0 ~ 255 之间，并看成是 16 模数，即除以 16 后取其余数。例：若算术表达式的值为 52。则其实际值为 52-INT(53 / 16) * 16 得 4。

颜色名称及其代号如下：

0	黑色	4	深绿色	8	棕色	12	绿色
1	红色	5	灰色	9	橙色	13	黄色
2	深蓝色	6	中蓝色	10	灰色	14	绿蓝色

3　紫红色　　7　浅蓝色　　11　粉红色　　15　白色

上例中 COLOR =52 实际表示设置深绿色。

3.7.4　PLOT

执行型: 立即与暂缓型

格式: PLOT 算术表达式 1, 算术表达式 2

功能: 在低分辩率图形方式下画一个色点。

说明: 算术表达式 1 为 X 坐标点, 其值范围在 0 ~ 39 之间; 算术表达式 2 为 Y 坐标点, 其值范围在 0 ~ 39 之间; 算术表达式 2 为 Y 坐标点, 其值范围在 0 ~ 47 之间。图形原点 (0, 0) 在屏幕的左上角上。

3.7.5　HLIN

执行型: 立即与暂缓型

格式: HLIN 算术表达式 1, 算术表达式 2　AT　算术表达式 3

功能: 在低分辩率图形方式下, 画出一条水平色线。

说明: 从坐标点 (算术表达式 1 的值, 算术表达式 3 的值) 到另一坐标点 (算术表达式 2 的值, 算术表达式 3 的值) 画一条直线。坐标点的范围与 PLOT 语句相同。

3.7.6　VLIN

执行型: 立即与暂缓型

格式: VLIN 算术表达式 1, 算术表达式 2　AT　算术表达式 3

功能: 在低分辩率图形方式下, 画出一条垂直色线。

说明: 从坐标点 (算术表达式 3 的值, 算术表达式 1 的值) 到另一坐标点 (算术表达式 3 的值, 算术表达式 2 的值) 画出一条直线。坐标点的范围与 PLOT 相同。

3.7.7　HGR

执行型: 立即与暂缓型

格式: HGR

功能：把屏幕设置为高分辨率第一页图形方式，并在屏幕下面留4行作为文本窗口。

说明：学习机具有高分辨率图形方式，在此方式下，一幅画面是由280 × 192个色点组成．高分辨率图形有两页，第一页的内存地址在＄2000～＄3FFF，第二页的内存地址在＄4000～＄5FFF。

执行HGR后，将屏幕清为黑色，并将原先在第一页图形区的图形清除。但该语句不影响HCOLOR的值。由于这个语句把屏幕下方4行设置为文本窗口，故一页画面由280 × 160个色点组成。要使屏幕设置为全屏幕图形方式，应在执行HGR之后加上POKE-16302，0或POKE 49234，0，若需再返回混合方式则用POKE -16301，0或POKE 49235，0。屏幕原点在屏幕的左上角上。

3.7.8 HGR2

执行型：立即与暂缓型

格式：HGR2

功能：把屏幕设置为全屏幕高分辨率第二页图形方式。

说明：执行此语句后，屏幕清为黑色，并将原第二页图形区的图形清除。但不影响CHOLOR的值。这是全屏幕图形方式，故有280 × 192个色点。要使屏幕设置为混合方式可用POKE-16301，0，但留出的4行文本窗口不能立刻作为程序文本使用，必须通过某种技术才能使用这个窗口。

3.7.9 HCOLOR

执行型：立即与暂缓型

格式：HCOLOR＝算术表达式

功能：在高分辨率图形方式下，用HCOLOR来设置颜色进行绘图。

说明：算术表达式的值范围在0～7之间。颜色名称及其代号如下：

0 黑1　　4 黑2

1 绿　　5 橙

2 紫　　6 蓝

3　白1　　　7　白2

HCOLOR 的颜色与 X 轴坐标的奇、偶数有关，另外还与显示器有关。

以 HCOLOR =3（白色）来说，如果 X 轴坐标为偶数，则可能显示出紫色；如果 X 轴坐标为奇数，则可能显示出绿色。只有当 X 轴紧邻的奇、偶点都显示时，才呈现白色。

HCOLOR 只能用在高分辩率图形方式下，不影响低分辩率图形的颜色。同理 COLOR 不影响高分辩率图形的颜色。

3.7.10　HPLOT

执行型：立即与暂缓型

格式：HPLOT 算术表达式 1，算术表达式 2

　　HPLOT　T O 算术表达式 3，算术表达式 4

　　HPLOT 算术表达式 1，算术表达式 2　T O 算术表达式 3，算术表达式 4〔{T O 算术表达式，算术表达式}〕

功能：在高分辩率图形方式下绘图。

说明：在第一页形式中坐标点（表达式 1 的值，表达式 2 的值）画出一个色点，颜色由 HCOLOR 表设置。

第二种形式是由当前坐标点到表达式 3，表达式 4 所指出的坐标点画出一条直线。如果执行这个语句之前没有画过点，则这条命令不会画出线来。

如果采用第三种形式，可根据各坐标点的前后次序依次画出一系列直线，而这些线条是连续的。使用这条命令最多只能有 239 个字符。X 坐标值（即奇数表达式）在 0 ～ 279 之间，Y 坐标值（即偶数表达式）在 0 ～ 191 之间，若所画点超出这个范围，则会造成错误？ILLEGAL QUANTITY ERROR。

例：下面这个程序将在高分辩率第一页图形区，分别以六种颜色（黑色除外）随机画出 30 条色线。

```
5   HOME
10   FOR HC = 1 TO 7
20   HGR
25   HCOLOR= HC
30   IF HC = 4 THEN 100
```

```
40  HPLOT 3,6
45  VTAB 21: PRINT "HCOLOR=";HC
50  FOR I = 1 TO 30
60 X =  RND (1) * 280
70 Y =  RND (1) * 192
80  HPLOT  TO X,Y
90  NEXT I
95  FOR J = 1 TO 500: NEXT
100  NEXT HC
```

如果要在高分辨率第二页图形区画出这些色线，则只需将 20 程序行改为 HGR2 即可。

3.7.11 MUSIC

执行型：立即与暂缓型

格式：MUSIC 算术表达式 1，算术表达式 2

功能：定义一个音阶和音长，并使机内扬声器发音。

说明：算术表达式 1 的值为发声的频率(即音高)，算术表达式 2 的值为发声的时间，它们的取值范围均在 0 ～ 255 之间。音阶与音长对应取值参考表如下，另外用户还可根据这个表及音响自行定出所需的音高。

X 值	255	288	205	192	171	152	140	128	114	102
音阶	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$	1	2	3	4	5	6	7

X 值	95	84	75	68	62
音阶	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$

Y 值	30	70	110	160	255
音阶	1/4 拍	1/2 拍	1 拍	2 拍	4 拍

例：一个发出音阶旋律的程序可按如下方法来设计。

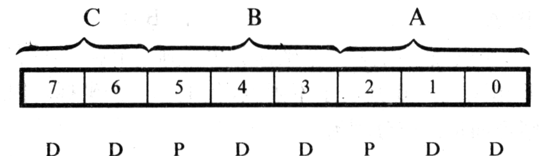
```
10  FOR I=1 TO 15
```

```
20  READ X,Y
30  MUSIC X,Y
40  NEXT I
50  DATA  255,160,228,160,205,160,192,160,171,
        160,152,160,140,160,128,160,114,160,102,
        160,95,160,84,160,75,160,68,160,62,160
```

3.8 高分辩率图形的向量作图法

CEC- BASIC 在高分辩图形区里有 5 个处理向量图形的语句: DRAW,XDRAW,ROT,SCALE 及 SHLOAD。在使用这些语句之前,应先用"图形向量"把图形定义出来,每一个图形都可以看成是一系列绘画向量的集合,这些集合放入内存中,一组图形中可以有一个或多个图形,这样一组向量图形与它们的索引构成"向量图形表",这些表可由键盘来建立,并存放于磁盘或磁带内以供使用。

由于定义的向量图形存放在内存中,每个字节(内存单元)由八位二进制数组成,因此每个字节分为三段,每段指定一个绘图向量:要不要画点,以及向哪个方向移动(上、下、左、右)。每个字节分配如下:



其中 0 ~ 2 位为 A 段,3 ~ 5 位为 B 段,6 ~ 7 位为 C 段。每段中有两位 DD,它指定一个移动的方向;在 A,B 段各有一位 P,它表示在移动之前要不要画一个点。对 D 与 P 的规定为下。

DD = { 00 向上移; 01 向右移; 10 向下移; 11 向左移 }

P = { 0 不画点; 1 画点 }

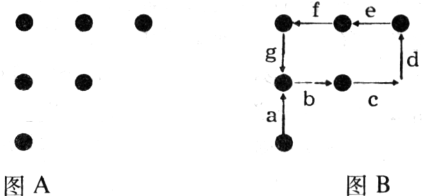
在C段中没有P栏，其自然值P=0，所以C段只能移动不能画点。在把各段组成“图形向量”时应遵守下面的规定：

①绘图时，先将图形绘成一个个向量点。这些图形向量存入内存时，先填A段，然后B段，最后C段。一个字节填完后，再填下一个字节。

②对每一个“图形向量”来说，如果C=00，则表示C段不起作用，不向任何方向移动。如果C=00，B=000则表示C.B两段都不起作用，只有A段起作用。

③如果A，B，C，三段全为0，则表示一个图形的结来。

例：要画出下图(A)中6个点组成的图形，可以用下图(B)中表示的a ～ g七个步骤来完成：



图A　　　　　　　　图B

其中a表示画一个点，然后上移一个坐标；b表示画一个点，然后右移一个坐标；c与b相同。但d则表示不画点，仅上移一个坐标。因此，a ～ g对应的图形向量段应分别为：

a: 100　　　　e: 111

b: 101　　　　f: 111

c: 101　　　　g: 110

d: 000(或00)

	C段	B段	A段	十六进制数
向量1	00	101	100	$ 2c
向量2	00	000	101	$ 05
向量3	00	111	000	$ 38
向量4	00	110	111	$ 37
向量5	00	000	000	$ 00

根据上述图形向量段组成“图形向量”，并在最后一个图形向量后面加一个全0向量作为图形结束，如上图所示。

如果有若干个图形，则先将图形按上述方法定义好，然后再加一个索引表，构成一个完整的图形表，一个图表形表最多可定义256个图形。一个完整的图形表的结构如下：



若将前面那个图形定义构造一个图形表，它应为下图

地址	内容	说明
S	$ 1	只有一个图形
S+ 1	$ 0	偏移量：S+ 4−S =4
S+ 2	$ 4	
S+ 3	$ 0	
S+ 4	$ 2C	图形定义
	$ 05	
	$ 38	
	$ 37	
	$ 00	

有了图形表后，下一步应把图形表送入机器中，在把表数据送入机器时，应确定表的首地址(S)的值。这个地址要选得合适，否则有可能破坏其它程序或表自身，一般首地址取 $ 6000 比较安全。以后将着重介绍如何将向量图形表存入内存的方法。

当使用这些向量图形时，应先把图形表的首址(S)存放到 $ E8 和 $ E9 单元中($ E8 存放低位，$ E9 存放高位)，然后采用下面的语句把定义的图形显示出来。

3.8.1 DRAW

执行型：立即与暂缓型

格式：DRAW 算术表达式 1　AT　算术表达式 2，算术表达式 3

　　　DRAW 算术表达式 1

功能：根据某个图形表在屏幕上进行作图。

说明：如果 DRAW 语句采用第一种形式，以某点(该点由算术表达式 2 的值与算术表达式 3 的值确定)为起始位置，画出一个图形。而所画的图形根据算术表达式 1 的值确定，该值表示画出图形表中的第几个图形。

算术表达式 1 的值必须在 0 到 n 之间，其中 n 是图形定义的个数(即图形表中首址 S 的内容)。算术表达式 2 的值在 0 ~ 279 之间，算术表达式 3 的值在 0 ~ 191 之间。如果 DRAW 语句采用第二种形式，则表示在最近执行 HPLOT.DRAW.XDRAW 的位置画出一个指定图形。

如果执行 DRAW 语句时，在内存中没有图形表，将会使系统僵死。

3.8.2 XDRAW

执行型：立即与暂缓型

格式：XDRAW 算术表达式 1〔AT 算术表达式 2，算术表达式 3〕

功能：用 DRAW 画图形颜色的补色画一图形。

说明：这个语句与 DRAW 一样可以画出一个图形，所不同的是用补色画图。如黑色与白色、蓝色与绿色，紫色与橙色是互为补色，因此使用 XDRAW 可擦掉屏幕上原有的图形。

3.8.3 SCALE

执行型：立即与暂缓型

格式：SCALE＝算术表达式

功能：决定 DRAW 与 XDRAW 所画图形的比例尺寸。

说明：算术表达式的值在 0 ～ 255 之间，SCALE=1 图形尺寸最小，数值越大比例尺越大，但 SCALE=0 表示最大的图形尺寸。

3.8.4 ROT

执行型：立即与暂缓型

格式：ROT＝算术表达式

功能：设置新画图形的旋转角度。

说明：旋转角度由算术表达式确定，其值在 0 ～ 255 之间。

ROT=0 不旋转。ROT=16 顺时针方向旋转 90 度。ROT=32 转 180 度，以此类推。ROT=64 与 ROT=0 相同，不旋转。这个语句与 SCALE 的值有关。当 SCALE=1 时，只有 4 个旋转值(即 0、16、32、48)。当 SCALE=2 时，有 8 个旋转值(即 0、8、16、24、32、40、48、56)，以此类推。计算机对不认识的旋转值，通常以较小而认识的旋转值计算。

如何将图形表送入内存，及图形的移动、旋转，下面结合前面所举的那个图形表为例：

把图形表送入内存一般有两种方法：

①利用 POKE 语句把图形表数据送入内存。如果图形表的首地址为 $ 6000(十进制 24576),则可把下面的程序输入计算机。

```
100   S=24576
200   FOR I=0 TO 9
300   READ  A
400   POKE S+I,A
500   NEXT I
600   DATA 1,0,4,0,44,5,56,55,0
```

此外,还需把这个首地址的十六进制数送入 232 和 233 单元中,可以由下面两条命令实现:

```
POKE 232,0
POKE 233,96
```

首地址 24576 的十六进制数可以这样求得:Y1=INT(X/256),Y2=X-Y1*256 其中:X 为十进制首地址,Y1 是十六进制高位数的十进制表示法,Y2 是十六进制低位数的十进制表示法,根据上述公式可求得高位数 96,低位数 0,然后把低位数存放在 232 单元,高位数存放 233 单元。

②利用监控状态将图形表存入内存:

首先键入 CALL- 151 进入监控状态,然后按下面的格式把数据输入到计算机里。

```
6000: D1 00 04 00 2C 05 38 37 00
```

最后把首地址存入 $ E8 和 $ E9 单元。$ E8 放低位,$ E9 放高位。即:E8:00 60。利用上述方法就将图形表数据放入计算机内存里了,接下去就可使用 DRAW、XDRAW、SCALE、ROT 等语句显示出图形,并可使图形动来。

例:

```
10  POKE 232,0: POKE 233,96
20  HGR2 : ROT= 0: SCALE= 1
30  FOR I = 10 TO 200: HCOLOR= 3:
       DRAW 1 AT I,100: HCOLOR= 0:
       DRAW 1 AT I,100: NEXT I
```

3.8.5 SHLOAD

执行型：立即与暂缓型

格式：SHLOAD

功能：把磁带中的图形表数据存入计算机内存。

说明：向量图形表存在 HIMEM：的正下方，然后自动将 HIMEM：设置在图形表的下方，可以保护图形表。同时图形表的起始地址会自动地给 CEC-BASIC 的图形绘制程序。如果有第二个图形表要进入内存，并代替第一个图形表，则在存入之前应先把 HIMEM：重新设置为原先的值，以免浪费内存空间。

把向量图形表存入磁带的方法：

①记下图形表的起始地址。

②记下图形表的尾地址。

③算出图形表所占内存的长度。②减①加上的值为长度。

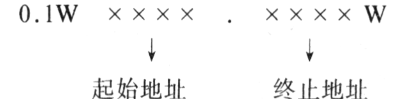
然后在监控状态下键入

0: ×× ××

↓ ↓

长度高位 长度低位

回车后，就可把图形表的长度，图形表的起始地址 x 终止地址键入：



当把磁带机调整好，按下磁带机的 RECORD 与 PLAY 键后，再按计算机的回车键，就可把数据存入磁带机。

若要以磁带机中的图形表数据存入内存，应先把磁带回到开始地方。键入 SHLOAD，按下磁带机 PLAY 键的同时按计算机的回车键。在听到两次“嘟”声后就表示图形表已装入内存。

3.9 函数

在 CEC-BASIC 中设有很多函数，用户可利用这些函数表处理程序中的各类问题。BASIC 函数可分为两大类：一类为标准函数，另一类为自定义函数。自定义函数已在前面介绍过了。对于标准函数又可分四类：一是三角函数；二是算术函数；三是字符串函数；四是其他函数。

下面介绍的这些标准函数可直接使用在 BASIC 程序中，也可作为立即执行方式中使用。

3.9.1 三角函数

SIN(算术表达式)　　　　　　正弦函数

其函数值等于算术表达式的值(弧度)的正弦。

COS(算术表达式)　　　　　　余弦函数

其函数值等于算术表达式的值(弧度)的余弦。

TAN(算术表达式)　　　　　　正切函数

其函数值等于算术表达式的值(弧度)的正切。

ATN(算术表达式)　　　　　　反正正切函数

其函数值等于算术表达式的值(弧度)的反正切。

以上这些都是三角函数，自变量均以弧度为单位，若自变量的值以度作单位时，要将度化为弧度制。具体的方法是(角度数 · π)／180。

3.9.2 算术函数

INT(算术表达式)　　　　　　取整函数

其函数值等于或大于算术表达式的值的最大整数。

例: INT(36.952) =36

INT(−7 * 1.1+ 6.1−9) =−11

这个函数用途很广，现归纳有以下几点:

· 可以对任意实数的小数部分进行四舍五入的处理，若实数为 X，则函数 INT(X+ 0.5)的值便是将 X 的小数部分进行四舍五入处理后的值。

例: 当 X=0.51 时　INT(X+ 0.5) =1

当 X=−0.8 时　INT(X+ 0.5) =−1

当 X=8.2　时　INT(X+ 0.5) =8

· 可求出任一整数的某位数字。

对任意的三位数 N，求各位数字的公式如下(位数变化时，公式也类似)。

百位数字 =INT(N／100)

十位数字 =INT((N−INT(N／100) * 100)／10)

=INT(N-(百位数字 * 100) / 10)

个位数字 =N-INT(N / 10) * 10

· 保留 N 位小数。

若有一个数为 X,将其保留 N 位小数,则可进行下式处理:INT(X * 10 ∧ N) / 10 ∧ N。

若需对 X 保留 N 位小数,且要对 N+ 1 位数进行四舍五入时则为:INT(X+ 10 ∧ N+ 0.5) / 10 ∧ N

· 若需判断 X 是否为整数时,可采用如下逻辑表达式 INT(X) =X,若表达式为真,则 X 是整数;若表达式的值为假时,则 X 不是整数。

RND(算术表达式)　　　　　随机函数

等于一个随机数,其值视表达式的值面有下述区别:

· 表达式的值大于零,产生 0 ~ 0.999999999 之间的随机数。

· 表达式的值等于零,产生一个与上次使用该函数所产生的数相同的随机数,它不受 CLEAR 及 NEW 的影响。

· 表达式的值小于零,产生一个与表达式的值一一对应的数,即相同的算术表达式的值,则产生相同的数。

例:

```
10   FOR I = 1 TO 10
20   PRINT  RND (1)
30   NEXT I
40   FOR I = 1 TO 5
50   PRINT  RND (0)
60   NEXT I
70   FOR I =  - 10 TO  - 5
80   PRINT "RND (";I;")="; RND (I)
90   PRINT "RND (";I - 1;")="; RND
       (I - 1)
100  NEXT I
]RUN
.505128773        .805846141
.402839286        .621381675
.984544252        .87997108
.126819831        .279336549
.538697286        .998197424
```

```
.998197424
.998197424
.998197424
.998197424
.998197424
RND  (-10)=3.73729563E-08
RND  (-11)=4.10982466E-08
RND  (-9)=3.3647666E-08
RND  (-10)=3.73729563E-08
RND  (-8)=2.99223757E-08
RND  (-9)=3.3647666E-08
RND  (-7)=5.2273208E-08
RND  (-8)=2.99223757E-08
RND  (-6)=4.48226274E-08
RND  (-7)=5.2273208E-08
RND  (-5)=3.73720468E-08
RND  (-6)=4.48226274E-08
```

由于随机函数所产生的是一个小于 1 的数，若需产生〔A，B〕的随机数，就可采用下述公式 RND(1) * (B−A)+ B。

若需产生〔A，B〕之间的随机整数时，可采用下公式： RND(1) * (B−A+ 1)+ B。

随机函数也是一个非常有用的函数，它常用于模拟一个过程；设计教学应用程序中随机产生题目供学生练习，在调试程序中也常使用。

SGN（算术表达式）　　　　　符号函数

如果算术表达式的值 > 0，则函数为 −1；如果表达式的值 =0，则函数值为 0；如果表达式的值>0，则函数值为 1。

即 $SGN\ (X)=\begin{cases}-1 & 当\ X<0\\ 0 & 当\ X=0\\ +1 & 当\ X>0\end{cases}$

ABS（算术表达式）　　　　　绝对值函数

其函数值是算术表达式的绝对值，即代数中的| X |。

SQR（算术表达式）　　　　　平方根函数

其函数值为算术表达式的值的平方根，算术表达式的值应大于或

等于 0。这个函数与代数中的$\sqrt{X}$一样。

EXP(算术表达式)　　　　　　指数函数

其函数值是以 e 为底，算术表达式的值为指数的函数(其中 e=2.718289)。

例:]PRINT EXP(1)

2.718289

LOG(算术表达式)　　　　　　对数函数

其函数值等于算术表达式的值的自然对数，即 LOGe(X)。根据对数函数中自变量的定义域范围，X 必须为正数。如果要计算以 10 为底的对数，则可用换底公式:

LOG(X) / LOG(10)

3.9.3 衍生函数

下面这些函数，在 BASIC 语言中虽不存在，但可用前面介绍的函数来计算，并能通过 DEF FN 的功能定义一个函数。

正割: SEC (X) =1 / COS(X)

余割: CSC (X) =1 / SIN(X)

余切: COT(X) =1 / TAN(X)

反正弦: ARCSIN(X) =ATN(X / SQR(-X * X+ 1))

反余弦: ARCCOS(X) =-ATN(X / SQR(-X * X+ 1))+ 1.5708

反正割: ARCSEC(X) =ATN(SQR(X * X-1))+ (SGN(X)-1) * 1.5708

反余割: ARCCSC(X) =ATN(1 / SQR(X * X-1))+ (SGN(X)-1) * 1.5708

反余切: ARCCOT(X) =-ATN(X)+ 1.5078

双曲正弦: SINE(X) =(EXP(X)-EXP(-X)) / 2

双曲余弦: COSH(X) =(EXP(X)+ EXP(-X)) / 2

双曲正切: TANH(X) =-EXP(X) / (EXP(X)+ EXP(-X)) * 2+ 1

双曲正割: SECH(X) =2 / (EXP(X)+ EXP(-X))

双曲余割: CSCH(X) =2 / (EXP(X)-EXP(-X))

双曲余切 COTH(X) =EXP(−X) / (EXP(X) −EXP(−X)) * 2+ 1

反双曲正弦: ARGSINH(X) =LOC(X+ SQR(X * X+ 1))

反双曲余弦: ARGCOSH(X) =LOG(X+ SQR(X * X−1))

反双曲正切: ARGTANH(X) =LOG((1+ X) / (1−X)) / 2

反双曲正割: ARGSECH(X) =LOG(ISQR(−X * X+ 1)+ 1) / X

反双曲余割: ARGCH(X) =LOG(SGN(X) * SQR(X * X+ 1) + 1) / X

反双曲余切: ARGCOTH(X) =LOG((X+ 1) / (X−1) / 2)

A MOD B: MOD(A) =INT ((A / B−INT(A / B)) * B+ 0.05 * SGN(A / B)

3.9.4 字符串处理函数

(1)LEN (字符串表达式)

这个函数可算出字符串内包含字符的总数,即字符串长度,其值在 0 ~ 255 之间。若串表达式中是几个字符串合并起来的,其长度不能超过 255。

例:
```
10  A $ ="ZHONG  HUA"
20  B $ ="XUE XI JI"
30  PRINT LEN(A $ + B $)
RUN
19
```

在中文状态下,由于一个汉字是由 3 个 ASCII 码组成,故中文字符长度为: 3 × (中文字个数)。

例:
```
PRINT LEN("中华学习机")
15
```

(2)STR $ (算术表达式)

这个函数将算术表达式的值转换成字符串。数值经过转换后,不能再参加算术运算。

例: PRINT STR $ (3+ 5)

```
8
PRINT STR $ (10 ∧ 10)
1E+ 10
```

(3) VAL(字符表达式)

这个函数将字符串表达式的字符串转换成数值，经转换后的这个数值可参加算术运算。如果字符串表达式的第一个为非空白字符，且不是数字的话，则函数值为 0。

例:

```
]PRINT VAL("1E10") / 1E10
1
]PRINT VAL("ABCD")
0
]PRINT VAL("   123")
123
]PRINT VAL("123ABC")
123
```

(4) CHR $ (算术表达式)

这个函数将算术表达式的值转换成与该值相对应的 ASCII 码字符，表达式的值必须在 0 ~ 255 之间。

例:

```
10  FOR I = 65 TO 90
20  PRINT I, CHR$ (I)
30  NEXT I

]RUN
65              A
66              B
67              C
68              D
69              E
70              F
71              G
72              H
```

73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

(5)ASC(字符串表达式)

这个函数将提供字符串表达式的第一个字符的 ASCII 码。

例:]PRINT ASC("ABC")

65

(6)LEFT $(字符串表达式,算术表达式)

这个函数将字符串从最左段取 X 个字符,X 的值由算术表达式决定。其中算术表达式的值在 1 ~ 255 之间。如果算术表达式值 >LEN(字符串表达式),就将字符串中的新字符取出,多余位置忽略。

例:]PRINT LEFT $("ABCDEF",4)

ABCD

]PRINT LEFT $("ABCDEF",10)

ABCDEF

(7)、RIGHT $(字符串表达式,算术表达式)

这个函数将字符串从最右段取 X 个字符,X 的值由算术表达式确定。其中算术表达式的值在 1 ~ 255 之间。如果算术表达式的值 > =LEN(字符串表达式)+ 1,就将字符串中的字符全部取出。

例:]PRINT RIGHT $("ABCDEF",4)

```
CDEF
]PRINT  RIGHT $ ("ABCDEF",10)
ABCDEF
```

(8)MID $(字符串表达式,算术表达式 1,〔算术表达式 2〕)

这个函数将字符串表达式中的字符从第 X 个字符开始,取出 Y 个字符,其中 X 的值由算术表达式 1 确定,Y 的值由算术表达式 2 确定。两个算术表达式的值应在 1 ~ 255 之间。如果两个算术表达式的值之和超过字符串总长度,则多余的位置不管。

若函数中无算术表达式 2,则将字符串表达式的字符串从第 X 个字符开始向左取到字符串尾,其中 X 的值由算术表达式 1 确定。下面这种情况,其结果是一样的。

MID $(字符串表达式,算术表达式)=RIGHT $(字符串表达式,LEN(字符串表达式)+ 1- 算术表达式)

例:
```
10  A $ ="ABCDEF"
20  PRINT MID $(A $,2,3)
30  PRINT MID $(A $,3,10)
40  PRINT MID $(A $,3)
RUN
BCD
CDEF
CDEF
```

由于中文状态中一个汉字由三个 ASCII 码组成,其格式为: $ 7F+ 区码 + 位码。

由于在 BASIC 程序中,所有的字符串在存储时最高位被屏蔽,因此其代码值在 $ 00 ~ $ 7F 之间。另外在 DOS 操作系统下,所使用的“,”和“:”作为变量与语句的分隔符,因此学习机器的区位码与国标中的代码有所区别。其区位码是通过转换的,它与国标中规定的区位码请见附录 5。

例如,汉字“啊”国标码为 1601,在 BASIC 程序中用字符串函数表示则为:A $ =CHR $(127)+ CHR $(46)+ CHR $(29)。

了解了汉字在计算机中的表示形式后,对使用 LEFT $,

RIGHT $ 及 MID $ 函数处理带有汉字的字符串时，应尤为谨慎，否则将截取不到你所需的子字符串。

例： 10　A $ ="中华学习机"

　　20　PRINT LEFT $ (A $,6)

　　30　PRINT RIGHT $ (A $,5)

　　RUN

　　中华

　　N.机

程序中 20 程序行从左截取 A $ 中 6 个字符，因此刚好截取到 2 个汉字"中华"。而 30 程序行实际是从左截取 5 个字符，结果得到"N. 机"，其中"N."是"习"的区码与位码 CHR $ (78) + CHR $ (46)。

3.9.5 其他函数

(1)TAB(算术表达式)

这是一个打印格式函数。使用这个函数可使光标往右移动，但不能在当前行上向左移，打印的内容将以表达式的值为起始列，开始打印。此函数必须用在 PRINT 语句中。算术表达式的值必须在 0 ~ 255 之间。如果表达式的值大于当前光标所在列，那么 TAB 会将光标移到规定的位置上，否则 TAB 失去作用。如果 TAB 的值为 0，则把光标移到第 256 个位置上。

例：

```
10   FOR I = 1 TO 5
20   PRINT  TAB( 6 - I);
30   FOR J = 1 TO 2 * I - 1
40   PRINT "*";
50   NEXT J
60   PRINT
70   NEXT I
```

```
]RUN

         *
        ***
       *****
      *******
     *********
```

(2) POS(表达式)

这个函数能得到当前光标所在列的位置。这里的表达式没有实际意义,但必须存在,一般可用 0 或 1 代替。如果光标在窗口的最左边就会得到 0。光标在窗口的最右边则得到 39。

(3) SPC(算术表达式)

这个函数一定要用在 PRINT 语句中。其功能是以上一次输出的最后一个字符位置为基准,跳过若干个空格,空格的格数由算术表达式的值确定。其中算术表达式的值应在 0 ~ 255 之间。几个 SPC 函数可以合起来,这样可以得到很大的空白位置。

例:]PRINT SPC(250) SPC(139) SPC(255)

(4) USR (算术表达式)

这个函数能将算术表达式传至机器语言的子程序中。其执行过程如下:

①将算术表达式的值 X 放进浮点累加器(即 $ 9D ~ $ A3 单元 X 中,注意不是累加器 A。

②转到 $ 0A 单元执行。通常在 $ 0A ~ $ 0C 中存放一个 JMP ×××× 指命,其中×××× 是用户自己设置的机器语言子程序的入口地址,而 $ 9D ~ $ A3 中的×值正是这个子程序的入口。

③在机器语言程序执行完毕返回 BASIC 时,必须在子程序结尾有一条 RTS 返回主程序指令,才能返回 BASIC。返回时,它把浮点累加器中的值作为函数的当前值。

例:
```
*0A :4C 00 03
* 0300 :60
* CTRL- C
] PRINT USR(8) * 3
24
```

在地址 $ 0A 开始放一个 JMP $ 300 指令,在 $ 300 地址放置一个 RTS 返回主程序指令,监控程序执行 JSR 指令到地址 $ 0A,而

$ 0A 是 JMP 跳转指令到 $ 300 又是 RTS 返回指令，回到 BASIC，因此函数返回的值就是以前放在累加器的 8，最后乘以 3 得到 24。

(5) FRE(算术表达式)

这个函数的功能可以释放数据区中的无用单元，使其返回可用变量的内存，这样扩大了变量的使用空间，并告诉用户还有多少变量可使用空间。函数中的表达式无实际意义，可设为 0 或 1。

程序在运行过程中，无用的字符串并不清除，对新的字符串则开辟新的内存单元。结果内存单元中保存了大量的废字符串，这样有可能破坏程序或入侵高分辨率图形区，若定期使用 X=FRE(0) 对内存进行大扫除，则可防止破坏性结果的产生。

例:
```
10  A $ ="中华":B $ ="学习机"
20  FOR I=1 TO 500
30  C $ =A $ + B $
40  NEXT I
50  PRINT PEEK(112) * 256+ PEEK(111)
60  X=FRE(0)
70  PRINT PEEK(112) * 256+ PEEK(111)
RUN
29876
37361
```

经 FRE(0) 函数处理后，清除 499 个废字串，释放出 7485 个内存单元。

(6) PDL (算术表达式)

这个函数能得到游戏控制器某号摇杆的当前值。函数值在 0 ~ 255 之间，这个值与控制器中的可变电阻有着一一对应的关系。算术表达式的值应在 0 ~ 3 之间，其表示某号摇杆。如果连续两个 PDL 读入二个游戏控制器的值，则在第二个游戏控制器上的读数，会受第一个的影响，为了准确起见，最好在两个 PDL 之间插入几个语句，或插入一个空循环。

例:
```
1000  X=PDL(0)
```

```
1010   FOR I=1  TO 10:NEXT
1020   Y=PDL(1)
```

除了用 PDL 可读出 4 个不同的游戏控制器的值外，还可用 PEEK 函数读出 3 个游戏按钮，其地址为 $ C061 ~ $ C063（或 $ C069 ~ $ C06B）十进制的 −16287 ~ 16285（或 −16279 ~ −16277）。

(7)SCRN（算术表达式 1，算术表达式 2）

在低分辨率图形方式下，使用 SCRN 会得到（算术表达式 1、算术表达式 2）位置上的十进制颜色代码。算术表达式 1 的值应在 0 ~ 39 之间，表达式 2 的值在 0 ~ 47 之间。

例:

```
10   HOME:GR
20   FOR I =1 TO 10
30   COLOR =INT(RND(1) * 16)
40   X=INT(RUN(1) * 40)
50   Y=INT(RND(1) * 40)
60   PLOT X,Y
70   VTAB 22:PRINT SCRN(X,Y)
80   GET A $
90   NEXT I
```

这个程序可在低分辨率图形方式下随机产生一些色点，这些色点的位置与颜色都是随机的，并在屏幕下方显示出该位置上色点的颜色代码。按任意键后，可产生下一个色点……。

(8)PEEK（算术表达式）

这个函数能读取以算术表达式的值为地址的内存单元的内容。所读取的数及地址均为十进制数，算术表达式的值应在 0 ~ 65535 之间。

例:

```
]PRINT  PEEK (−16287)
```

可读出第一个游戏按钮的当前值。

第四章
CEC-LOGO语言程序设计

LOGO 语言是美国麻省理工学院的西摩·佩帕特教授领导下的 LOGO 小组研制成功的一种高级计算机语言。它以认识心理学和人工智能原理为根据，目的是给少年儿童提供一种较好的智力开发环境，使儿童们不仅学会使用计算机，同时还学习如何象数学家、语言学家一样进行思考与解决问题。

LOGO 语言是一种过程性语言，即程序是由过程(procedure)组成的。所谓过程是为达到某种目的而按一定的规则组织起来的 LOGO 语言的语句集合，这些过程又可作为其它过程的子过程，组成更复杂的过程，以此来解决复杂的问题。因此 LOGO 语言具有模块化结构，每个程序可由许多相对独立的过程组成，使设计方便、灵活。过程一经定义便成为一个新的命令，以后可用来作为基本命令来调用，扩充性强。程序具有递归功能，给编制高级水平的程序带来方便。

LOGO 语言是一种接近自然语言的高级语言，只要认识一些英文单词的人就能方便地进行人机对话，非常简单易学。

LOGO 语言数据结构丰富，它不仅包含数和字符串，还包括表(LIST)的混合结构，能进行字表处理，模拟人工智能等。

LOGO 语言具有较强的绘图功能，非常形象直观，这也是 BASIC 语言所不及的。

LOGO 语言在 APPLE 计算机上也有多种版本，但这些版本的 LOGO 都要通过磁盘驱动器才能实现。因此对 LOGO 的推广普及是一大障碍。今天，CEC-I 型中华学习机将 LOGO 语言(CEC-LOGO)固化于主机电路板上的只读式存储器内，使用户直接享受 LOGO 的无穷乐趣。

CEC-LOGO 基本上具有 Terrapin LOGO 的功能。当系统已装有 DOS 操作系统的情况下，CEC-LOGO 的命令系统与

Terrapin LOGO 完全相同,而系统没有 DOS 时 CEC-LOGO 不能进行程序存取盘操作。

4.1 进入 CEC-LOGO 语言

CEC-I 型中华学习机虽固化有 LOGO 语言,但开机后系统首先进入 BASIC 语言状态,因此必须在 BASIC 状态下进入 LOGO 语言。

① 在系统未装有 DOS 的 BASIC 状态下,键入:

```
]LG
```

即可进入 LOGO 语言。

② 在系统已装有 DOS 的 BASIC 状态下,键入:

```
]MAXFILES1
]LOGO
```

即可进入 LOGO 语言。

系统进入 LOGO 语言后,屏幕马上出现下图,并出现 LOGO 语言提示符"?"

```
CHINESE EDUCATION COMPUTER
LOGO
VERSION 1.1
1 9 8 7.6
?
```

下面对 CEC-LOGO 的语句规则作一一介绍。

4.2 CEC-LOGO 的变量与语句规则

4.2.1 变量名

在 CEC-LOGO 中,变量可以表示数、词汇及表。变量用英文字母或数字来表示,用这些字符作为变量的名字,称为变量名。例: AB、A1、FD 等都可作变量名。注意: 一个变量名中不能有空格,例: A, B 就不

能作为一个变量名。在使用变量时，一般在变量名的前面都有一个符号作为前导，以后将作详细介绍。

4.2.2. LOGO数、表、词汇的表示法

CEC-LOGO 使用的均为十进制数。对纯整数运算 LOGO 能表示出10位有效数字，其数的范围在|X|≤2147483647。

例：求12345678+8765432的值，可键入：

PRINT 12345678+8765432

100000000

若进行实数运算 LOGO 只能表示出6位有效数字，如果一个数小于等于6位有效数字时，可用常规表示法。

例：?PRINT 65.786+123

188.786

如果一个数超过6位有效数字，而且绝对值大于零时，则用 S×.×××××ETT 表示。其中 S 为数的符号，正数无表示，负数用"-"表示，X 为0～9之间的10进制数字。E 表示10的幂，且指数的符号为正，TT 为指数值。

例：?PRINT 88888*88888

7.90121E11

其表示为88888*88888的值为$7.90121*10^{11}$。

如果一个数超过6位有效数字，且此数的绝对值小于零时，则用 S×.×××××ETT 表示。其中 N 表示为10的幂，且指数的符号为负，其他字母的表示与上相同。

例：?PRINT 0.0000130021

1.30021N05

即表示为$1.30021*10^{-5}$。

LOGO 除可表示数外，还可处理字句，字是一串字符，字符串中没有空格，LOGO 把这样的字称为"词"，例：TEACHER、STUDENT 等都为词。要在计算机上输出词，可用 PRINT 命令。

例：PRINT "STUDENT

STUDENT

注意在词的前面用双引号作为前导。

对数字词来说可以加双引号，也可以不加双引号，这也就是说数字词它具有双重身分，它即可代表一个词，也可代表一个数参加数学运算。

例: ?PRINT 10+ "10

20

LOGO 把由一系列空格隔开的字组成的句子称为"表"(LIST)。通常用一对方括号把表括起来。表中的字不能带双引号，字与字之间的空格起分隔作用。表中用空格分开的字或数都称为表的元素。要在屏幕上把一个表打印出来，也可用 PRINT 命令。

例: PRINT 〔ZHONG HUA XUE XI JI〕

ZHONG HUA XUE XI JI

有关词表的处理方法将在以后详细介绍。

4.2.3 LOGO 运算符与表达式规则

在 LOGO 中运算符可分为两类:

① 算术运算符:

+ (加)、—(减)、* (乘)、/ (除)

② 逻辑运算符:

<(小于)、>(大于)、=(等于)、ALLOF(与)、ANYOF(或)、NOT(非)

LOGO 完全可以当作一个非常高级的计算器来使用。数学表达式的规则与数学及 BASIC 语言相同，而且数学运算优先等级与教学中的规定一致。括号全部使用圆括号，并可以多层嵌套。

例: 计算5+ 3 * 4 =?，可键入:

?PRINT 5+ 3 * 4

17

再计算〔(5+ 3) * 4 —2〕* 2 =?，可键入:

?PRINT ((5+ 3) * 4 —2) * 2

60

LOGO 在运算时，也可不用 PRINT 命令，直接键入表达式，也能得到运算结果。

例: ? (6+ 3) / 3

RESULT: 3

另外 LOGO 中还有一些标准函数，用户也可在算术表达式中直接引用，这些函数将在以后介绍。

LOGO 中的逻辑运算符可对表达式进行运算如 =、<、> 三种符号可对两个算术表达式的值进行比较，凡经比较后的结果不再是数值，不能再参加数学运算。

例: ? 1 / 2 = 0.5

RESULT: TRUE （表示结果为真）

?4 + 4 = 9

RESULT: FALSE （表示结果为假）

而 ALLOF、ANYOF、NOT 三种运算符可对所有的表达式进行运算。

例: ?NOT 4 < 5

RESULT: FALSE

其它词、表的逻辑运算将在以后介绍。

最后说明一点：由于 LOGO 在进行数学运算时是将十进制数转换成二进制数，然后运算，最后再将二进制结果转换成十进制数。经这样转换后，有可能所得结果不十分准确。

4.2.4 命令行

一个命令行中可以是一个命令，也可以同时设置若干个命令，机器从左到右依次执行这些命令。每个命令之间一定要用空格分开，一个命令行最多输入 255 个字符，其中包括空格。例如：? DRAW FD 50 RT 120 FD 50 RT 120 FD 50 就可在屏幕上画出一个三角形，在这条命令行中包括了 6 条命令。

4.2.5 执行模式

直接模式：在屏幕上出现提示符"?"时，系统处于直接执行模式状态下，此时键入命令后，计算机马上执行，上例所画的三角形就是以直接执行模式键入的。

定义模式：若在直接模式状态下键入 TO ××× 后按回车键，计算机立即执行 TO 命令，使系统离开直接执行模式，进入定义模式。进

入定义模式后，提示符“?”消失，用户键入其它命令，计算机并不立即执行。当把程序中所有命令都键入后，可按CTRL－C脱离定义模式，完成定义，回到直接模式。经过定义的这些命令集合称为过程，过程一经定义，便被存放在计算机内存的工作区内，工作区内的过程可反复被调用。因此凡经过定义的过程，它就象命令一样，可以立即执行，也可以是其它过程中的一条命令。

在定义模式下，键入一个画圆的程序：

```
TO  DRAW
REPEAT  360 〔FD  1  RT  1〕
END
CTRL－C        (表示CTRL与C键时同时按下)
```

此时可发现，所键入的程序，机器不马上执行，而必须键入使程序运行的命令：

?DRAW

才可使机器执行键入的程序，画出一个圆。

若在定义模式状态下按CTRL－G也可返回直接模式，但此时是强迫中断定义过程。

编辑模式：一个定义好的过程，在使用EDIT命令后，就可使系统进入编辑模式，机器把一个过程显示在屏幕上，在屏幕的紧下方出现一行白底黑字：EDIT：CTRL0－C TO DEFING，CTRL－G TO ABORT。此时可使用编辑命令对过程体进行修改编辑，退出编辑模式可按CTRL－C或CTRL－G，随之系统回到直接执行模式。

4.3 LOGO画图命令

CEC－LOGO具有很强的作图功能，这也是广大青少年喜欢使用LOGO的原因之一。下面对作图方面的命令作一一介绍。

4.3.1 DRAW

执行型：立即与暂缓型

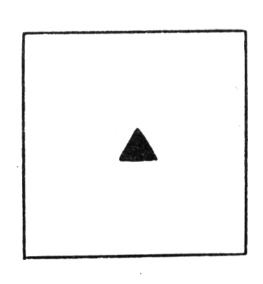
格式：DRAW

功能：使LOGO进入绘图模式然后清屏，使图标置于原点。

说明：在画一个新图形之前，首先要清屏，并将图标回到原点。此

时图标头朝上。

在绘图模式下整个屏幕以图标起始位置作为原点(0,0),如下图,第一个数表示横坐标,



共范围在 －140～139之间。第二各数字表示纵坐标,其范围在 －199～120之间。在执行DRAW命令后使屏幕设置为图形与文本混合模式,此时在屏幕下方有四行作为打入命令及显示文字数据的窗口,因此图标所作图形到此被遮掉了。

4.3.2 FORWARD

执行模式:直接与定义模式

格式:FORWARD <算术表达式>

缩写形式:FD <算术表达式>

功能:使图标向前移算术表达式的值个图步。

说明:算术表达式的值可正可负,若为正值,图标向前移动;若为负值,实际是向后移。该命令自动对算术表达式的值取整。

例:

```
?FD  50                    (向前移50步)
?FD  —50                   (向后移50步)
?MAKE  "A  50              (给变量A 赋值50)
?FD:A + 50 (向前移100步,其中:A + 50为算术表达式
            其值为100)
```

4.3.3 BACK

执行模式:直接与定义模式

格式:BACK <算术表达式>

缩写形式: BK　< 算术表达式 >

功能: 使图标向后移算术表达式的值个图龟步。

说明: 算术表达式的值为正, 表示向后移若干步; 若该值为负时, 实际是向前移若干步。该命令自动对算术表达式的值取整。

例:　?BK　5 0　　(向后移5 0 步)

4.3.4　LEFT

执行模式: 直接与定义模式

格式: LEFT　< 算术表达式 >

缩写形式: LT　< 算术表达式 >

功能: 使图标向左移算术表达式的值所指的度数。

说明: 算术表达式的值为正时, 图标向左转; 若该值为负数时, 则向右转; 此命令自动对算术表达式的值取整。

例:　LEFT　9 0　　(向左转9 0 度)

4.3.5　RIGHT

执行模式: 直接与定义模式

格式: RIGHT　< 算术表达式 >

缩写形式: RT　< 算术表达式 >

功能: 使图标向右移算术表达式的值所指的度数。

说明: 算术表达式的值为正时, 表示图标向右转; 若为负值时, 则向左转。机器自动对算术表达式的值取整。

例:　RT　9 0　　(向右转9 0 度)

4.3.6　PENCOLOR

执行模式: 直接与定义模式

格式: PENCOLOR　< 算术表达式 >

缩写形式: PC　< 算术表达式 >

功能: 根据算术表达式的值, 设置笔的颜色。各种颜色及其相对应的代码如下表:

代码	颜色	代码	颜色
0	黑色	1	白色

2	绿色	3	紫色
4	桔红色	5	蓝色
6	逆转		

说明:设置笔色的值一定要在0～6范围内,否则将出现 PC DOESN'T LIKE ×× AS INPUT。PENCOLOR 6 表示图标所绘的线条与底色互为补色,即白黑互为补色。当时同一线条在 PENCOLOR 6 状态下连画两次,会使线条恢复成底色。对不同的彩色监视器或电视机可能色彩有所不同。LOGO 在初始情况下,机器自动设置 PENCOLOR 1。

将下程序键入计算机,然后执行就可看到用不同颜色的笔所画的各种线条。

```
TO  PIC:COLOR
IF:COLOR =6  THEN  STOP
PENCOLOR :COLOR
REPEAT  10〔FD 100 RT 90 FD 1 RT 90
        FD 100 LT 90 FD 1 LT 90〕
PIC:COLOR + 1
END
CTRL  C                (CTRL 与 C 键同时按下)
PIC  1
```

4.3.7 BACKGROUND

执行模式:直接与定义模式

格式:BACKGROUND < 算术表达式>

缩写格式:BG < 算术表达式>

功能:设置屏幕背景颜色,即底色。其颜色范围的值为0～5,所对应的颜色如下表:

代码	颜色	代码	颜色
0	黑色	1	白色
2	绿色	3	紫色
4	桔红色	5	蓝色

说明:若所设屏幕背景色的值不在0～5范围内,则显示出 BG

DOESN'T LIKE ×× AS INPUT。LOGO在初始情况下，机器自动设置 BACKGROUND 的值为0，即背景为黑色。

例: ?DRAW

?BG 1 (设置背景为白色)

?BG 5 (设置背景为蓝色)

另外当先在屏幕上画好图形，再设置底色时，将影响所画图形的颜色。反之则不影响。

4.3.8 PENDOWN

执行模式：直接与定义模式

格式：PENDOWN

缩写格式：PD

功能：落笔，使图标在移动过程中绘图。

说明：计算机在进入 LOGO 后，自动设置成落笔状态，即在这之后图标移动时，都留下痕迹。

4.3.9 PENUP

执行模式：直接与定义模式

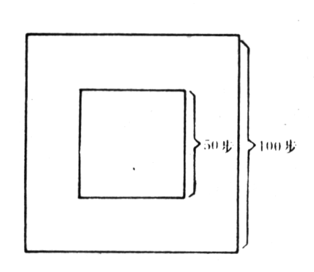
格式：PENUP

缩写格式：PU

功能：提笔，使图标在移动过程中不留下痕迹，即不作图。

说明：执行该命令后，所画的线条都不会在屏幕上出现，只有用 PENDOWN 命令才可屏幕上画出线条，所画线条取决于 PENCOLOR 的值。

例：要画下图的两个正方形，可用这些命令。



```
?PENUP
?FD 25
?PENDOWN
?RT 90 FD 25 RT 90 FD 50 RT 90
  FD 50 RT 90 FD 50 RT 90 FD 25
?PENUP
?LT 90 FD 25
?PENDOWN
?RT 90 FD 50 RT 90 FD 100 RT 90
  FD 100 RT 90 FD 100 RT 90 FD 50
```

4.3.10 HIDETURTLE

执行模式:直接与定义模式

格式:HIDETURTLE

缩写格式:HT

功能:使图标隐没

说明:若图形画好后,不再需要屏幕出现图标,此时可使用该命令,例如上例中当执行 HT 后,屏幕上只留下两个正方形,图标隐去.但应注意:图标虽然隐没,并不意味着不能画图,只是图标的朝向及当前点看不到。

4.3.11 SHOWTURTLE

执行模式:直接与定义模式

格式:SHOWTURTLE

缩写格式:ST

功能:显示出图标的形状。

说明:当图标隐没后,若再想把图标显示出来,可使用 ST 命令,执行该命令后,屏幕上立刻显示出当前图标的形状,以便于继续画图。

4.3.12 CLEARSCREEN

执行模式:直接与定义模式

格式: CLEARSCREEN

缩写格式: CS

功能: 清屏,不改变图标的状态。

说明: 这个命令不同于 DRAW 命令,图标并不回到原点(0,0)。

例: ?DRAW

?FD 50 RT 120 FD 50 RT 120 FD 50

?CS

可看到图标并不回到原点。

4.3.13 HOME

执行模式: 直接与定义模式

格式: HOME

功能: 移动图标于屏幕中心,标头指向上方。

说明: 这个命令也不同于 DRAW 命令,该命令虽使图标回到原点,但没有清屏的作用。另外当执行 HOME 使图标从当前点回到原点这段直线轨迹根据提落笔、及笔色的状态来决定是否画出。

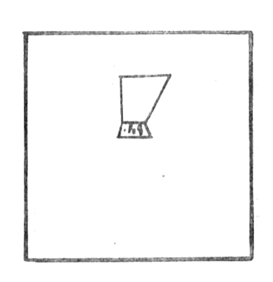
例: 执行下述命令:

?PD PC 1

?FD 100 RT 90 FD 100

?HOME

可看到下图形:



4.3.14 NOWRAP

执行模式:直接与定义模式。

格式:NOWRAP

功能:防止图标围绕屏幕作图。

说明:由于屏幕坐标:X是从 -140~139之间,Y是从 -119~120之间。当图标移动的距离过大时,就有可能把线条画出屏幕的一端,而又从屏幕的另一端画出来,这样可能造成破坏画面的后果。为了防止这种错误发生,可使用 NOWRAP 命令,防止上述情况发生。

例: ?NOWRAP

?FD 300

TURTLE OUT OF BOUNDS

由于图标前进的步数太大而出错,防止线条从屏幕上沿重新进入屏幕。

4.3.15 WRAP

执行模式:直接与定义模式

格式:WRAP

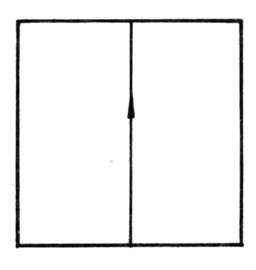
功能:使图标围绕屏幕作图。

说明:系统进入 LOGO 后,自动设置为 WRAP。

例: ?WRAP

?FD 300

执行这两个命令后,可在屏幕上看到图标。



从屏幕上沿穿出,又从下沿进入进行作图。

4.3.16 SETHEADING

执行模式:直接与定义模式

格式:SETHEADING < 算术表达式>

功能:根据算术表达式的值设置图标的朝向。

说明:有时希望将图标指向某一角度,与原有的指向无关,就可用该命令。使用时朝向为0 或3 6 0 度是指图标朝上,而朝向为1 8 0 度是指向下,以此类推。

例: ?DRAW

?SETHEADING 1 8 0 (图标指向朝下)

?SETHEADING 1 8 0 (图标指向仍朝下)

4.3.17 SETX

执行模式:直接与定义模式

格式:SETX < 算术表达式>

功能:水平移动图标到算术表达式的值所指的X 坐标。

说明:该命令可将图标移到X 轴向的某个坐标点上,而Y 坐标点不变。图标所经过的直线轨迹是否画出,视提落笔及笔色的状态。

HOME 位置的X 坐标为0,屏幕的右半部分X 坐标为正,而左半部分为负。因此 SETX 100 SETX − 100 与 FD 100 FD − 100 (假设此时图标朝向9 0 度)的结果是不一样的。

4.3.18 SETY

执行模式:直接与定义模式

格式:SETY < 算术表达式>

功能:重直移动图标到算术表达式的值所指的Y 坐标。

说明:该命令可将图标移到Y 轴向的某个坐标点上,而X 坐标点的位置不变。图标所经过的直线轨迹是否画出,视提落笔及笔色的状态而定。

在原点的上半部分Y 的值为正,而下半部分则为负。

4.3.19 SETXY

执行模式:直接与定义模式

格式　SETXY　< 算术表达式1 >　< 算术表达式2 >

功能　移动图标到算术表达式1，算术表达式2 的值所指定的 X，Y 坐标位置上。

说明：当算术表达式2 为负常数时，一定要在负数两边加圆括号。

例：SETXY 5 0 (－ 5 0)，此例与 SETX 5 0 SETY － 5 0 是等效的。

4.3.20　TOWARDS

执行模式：直接与定义模式

格式：TOWARDS　< 算术表达式1 >　< 算术表达式2 >

功能：输出图标当前位置到算术表达式1 的值，算术表达式2 的值所指定的 X，Y 坐标点的方向角。

说明：当算术表达式2 为负常数时，一定要在负数两边加圆括号。

输出方法有两种：

. 直接用 TOWARDS 命令。

例：
```
?HOME
?TOWARDS  1 0 0  1 0 0
RESULT:4 5
```

表示(1 0 0，1 0 0)到(0，0)点的方向角。

. 用 PRINT(输出命令)输出。

例：
```
?HOME
?PRINT  TOWARDS  1 0 0  1 0 0
?4 5
```

4.3.21　TURTLESTATE

执行模式：直接与定义模式

格式：TURTLESTATE

缩写形式：TS

功能：输出图标笔和颜色的状态表。

说明：执行该命令将输出四项内容。第一项为笔状态，落笔状态输出为真(TRUE)，提笔状态输出为假(FALSE)；第二项为图标是否显示，若显示图标状态为真(TRUE)，隐没图标为假(FALSE)；第三项为

屏幕背景颜色,其输出值为0 ~5;第四面为笔的颜色,其值为0 ~6。该语句的输出方法也有两种:

. 直接使用 TURTLESTATE 命令。

?TS

RESULT:〔TRUE TRUE 0 1〕

表示落笔。显标、背景为黑色及笔色为白色。

. 使用 PRINT 命令输出

?PRINT TS

〔TURE TURE 0 1〕

4.3.22 XCOR、YCOR

执行模式:直接与定义模式

格式:XCOR

YCOR

功能:输出当前图标X 坐标或Y 坐标值。

例: ?HOME

?XCOR

RESULT:0.

?YCOR

RESULT:0.

或?HOME

?PRINT XCOR

0.

?PRINT YCOR

0.

4.3.23 .ASPECT

执行模式:直接与定义模式

格式:.ASPECT < 算术表达式>

功能:设置屏幕垂直标尺因子,以补偿显示器或电视机屏幕上圆形,正方形等被压扁的现象。

说明:系统进入 LOGO 时,机器自动设置.ASPECT 0.8。

例: ?. ASPECT 1.0

可发现此后所画图形的纵向步长比横向步长一点,从而使图形的纵向拉长。

因此用户在选定显示设备后,选择一个合适的. ASPECT 数值。另外该补偿命令必须用在画图形之前,而不能在画图之后。

4.3.24 FULLSCREEN

执行格式: 直接与定义模式

格式: FULLSCREEN

功能: 执行该命令后,使屏幕设置为全屏幕作图模式,即整个屏幕上图标绘的图形都能显示出来。

4.3.25 SPLITSCREEN

执行模式: 直接与定义模式

格式: SPLITSCREEN

功能: 执行该命令后,使屏幕设置为文本图形混合模式。屏幕上半部分为图形区,下方有四行为文本显示区。

说明 系统进入 LOGO 作图时,机器自动设置为 SPLITSCREEN 状态。

4.3.26 TEXTSCREEN

执行模式: 直接与定义模式

格式: TEXTSCREEN

功能: 使屏幕设置为文本显示模式。

4.3.27 NODRAW

执行模式: 直接与定义模式

格式: NODRAW

缩写形式: ND

功能: 退出图标作图,清屏; 进入文本模式,光标移到屏幕的左上

角。

4.4 过程及其编辑命令

LOGO 语言可以定义过程，所谓过程就是程序。前面介绍的这些命令，所例举的基本上都是直接执行模式，即一个命令行输入完后，按一下回车键，计算机立即执行该命令行，随之命令也即消失。如果将这些命令行定义成一个过程，并给这个过程取一个“过程名”，这样被定义后的“过程名”就可作为新的 LOGO 命令，以后可以多次被引用，而不须重键入命令行。

下面对如何定义过程及过程的编辑方法介绍如下。

4.4.1 TO

执行模式：直接与定义模式

格式：TO < 过程名 >

TO < 过程名 > < 参数表 >

功能：定义一个过程。

说明：过程名是由英文字母或数字组成，中间不能断开，即过程名中不能有空格；原语命令不能作为过程名。

例： ?TO FD

FD IS A LOGO PRIMITIVE

机器指出 FD 是 LOGO 的原语命令，不能作为过程名。

当键入 TO < 过程名 > 并回车后，计算机立即执行 TO 命令，使计算机进入定义模式。进入定义模式后，提示符“?”消失。接着可键入其它命令，但不立即执行。一个过程的形式为：

TO < 过程名 >

.
.
.

< 过程体 >

.
.
.

END (END 作为过程的结束符)

当把过程键入后，按 CTRL- C 键在屏幕上将显示出：

< 过程名 >　DEFINED

?

表示系统完成了一个过程的定义，并离开定义模式返回到直接执行模式。此时"?"重新出现。如果键入 < 过程名 > 后回车，计算机就立即执行 < 过程名 > 指出的过程内的命令，这就叫过程的调用。

例：
```
TO  BOX            (BOX 为过程名)
 FD  50  RT  90  FD  50  RT  90
 FD  50  RT  90  FD  50
 END
 CTRL-C            (完成定义)
 ?BOX
```

立即执行这个新的 LOGO 命令，画出一个正方形。

过程定义的第一种形式中没有 < 参数表 > 这一项，称为无参数过程。而第二种形式有一个 < 参数表 >，在 < 参数表 > 中是一系列用空格隔开的参数名，这些参数名实际为变量名，在这些变量名前通常都有一个":"作为引导。在定义好一个过程后，只要用 < 过程名 > < 参数值表 >，即可用 < 参数值表 > 内的数据一一代换掉 < 参数表 > 中所用到的相应位置的参数。

例：
```
TO  ZFX :CH
FD :CH  LT  90
FD :CH  LT  90
FD :CH  LT  90
FD :CH
END
CTRL-C
  ?ZFX  50
```

此后计算机调用 ZFX 这个过程，并将50代替过程体中的:CH，从而画出一个边长为50的正方形。而

  ?ZFX 100

则画出一个边长为100的正方形。

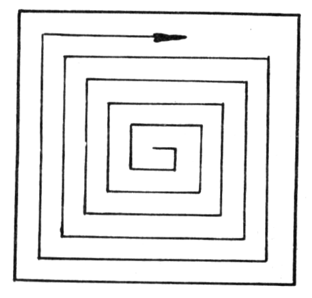
过程体中的语句可以是任何命令行，也可以是过程调用。在过程体内也可调用同名过程，即过程调用它自身，这称为递归调用，这种过

程称为递归过程。

例: 递归螺旋形过程

```
TO  SPI  :LENGTH
FD  :LENGTH  RT  90
FD  :LENGTH  RT  90
SPI  :LENGTH+ 5
END
CTRL-C
?SPI  5
```

可画出下图形,它将一直画下去,如果要停止这样一个无终止的程序执行过程,可使用 CTRL-G,强迫中断。



< 参数表> 中的参数也可以不止一个,但必须两个参数中间用空格隔开,而且对负参数两边一定要用圆括号括起来。把上例中递归增加的步长也设为参数,则过程体变为:

```
TO  SPI  :LENGTH  :INC
FD  :LENGTH  RT  90
FD  :LENGTH  RT  90
SPI  :LENGTH + :INC  :INC
END
```

在调用此过程时也必须输入两个参数值。

```
?SPI  5  10
   ⋮
```

4.4.2 END

执行模式: 定义模式

格式: END

功能: 结束过程定义。

说明: 此命令只能用在定义模式中,若在直接执行模式中使用,则会出现 END SHOULD BE USED ONLY IN THE EDITOR。

4.4.3 EDIT

执行模式: 直接执行模式

格式: EDIT < 过程名>

　　　EDIT

缩写格式: ED < 过程名>

　　　　　ED

功能: 使系统进入编辑模式(与定义模式相类似),过程的内容在屏幕上列出来,光移到首行内容的最左端,此时可用其它编辑键对过程行进行修改和编辑。

说明: 第一种形式将对所指定的过程进行编辑; 第二种形式将对最新键入的过程进行编辑。

以下命令均在编辑模式上使用,供修改过程及改正错误用。

4.4.4 ◁ 与 ▷ 键

这两个键分别能使光标所在的当前行的光标向左移或向右移,而不改变任何字符。在行的端点如继续按这两个键会使光标移向另一行,当光标移到首行第一个字符,再继续按 ◁ 键,则光标不动,并响警铃; 同样当光标移到过程体最后一行末尾,再继续按 ▷ 键,则光标也不动,并响警铃。

4.4.5 CTRL-D

删除光标下的一个字符。

例:

```
BK  50
     ↑
```

光标

按 CTRL-D，会出现

BK　0

↑

光标

再键入3，则变为:

BK　3 0

↑

光标

因此若需在一行内插入一些命令，可先用光标移动键移至插入处，然后键入所插入的命令即可。

4.4.6　CTRL- C

退出编辑模式，定义被编辑的过程，系统回到直接执行模式。切切注意：凡新输入的过程或对过程进行编辑后一定要按 CTRL- C。

4.4.7　CTRL- G

退出编辑模式，对原过程不作任何修改（不论在编辑模式下，对此过程作过何种修改都作废），回到直接执行模式。

4.4.8　Esc 键

删除光标左边一个字符，同时光标向左移一格。

例:　FD　1 0 0■

↑

光标

按一次 Esc 键后会出现:

FD　1 0■

↑

光标

4.4.9　CTRL- P

光标上移一行。当光标移到首行后，再按 CTRL- P 则响警铃。

例: FD 50

LT 90

↑

光标

按一次 CTRL- P 则会出现:

FD 50

↑

光标

LT 90

4.4.10 CTRL- N

使光标下移一行。当光标移到过程体末尾行,再按 CTRL-P,则光标不动,同时响警铃。

上例中再按一次 CTRL- N,则可出现:

FD 50

LT 90

↑

光标

4.4.11 CTRL- O

执行这个命令可在两行中间出现一空行,供插入新行之用。

若想在 FD 100

LT 45

这两个过程行中插入一新行,其方法为先将光标移到 LT 45 的第一个字符位置,即

LT 45

↑

光标

然后按 CTRL- O 后出现:

FD 100



↑

光标

LT 45

此后可在新打开的这一行键入命令，然后回车，这样完成插入一新行。

使用此命令应特别注意光标一定要移到过程行的最左端，否则不会打开一新行。

例：FD 100

LT 45

↑

光标

按 CTRL- O 后则出现

FD 100

LT ■

45 ↑

光标

4.4.12 CTRL- A

使光档移到当前行行首。

例：

.
.
.

FD 50

↑

光标

.
.
.

按 CTRL- A 后则出现

.
.
.

FD 50

↑

光标

.
.
.

4.4.13 CTRL-E

使光标移到当前行行尾。

上例中按 CTRL-E 后则出现:

```
      .
      .
      .
 FD  50
       ↑
     光标
      .
      .
      .
```

4.4.14 CTRL- K

删除当前行中光标以右的全部字符。

例:
```
FD  55000
      ↑
     光标
```

按 CTRL- K 后则出现:

```
 FD  5■
      ↑
     光标
```

4.4.15 CTRL- B

使光标向前(即向过程开始方向)移一页,若过程不足一页则移到开始处。

4.4.16 CTRL- F

使光标向后(即向过程尾方向)移动一页,若不足一页则把光标移到过程最后一个命令行上。

4.4.17 CTRL- L

使光标移到屏幕的中央。

4.5 控制过程的执行及流程命令

这部分命令可改变程序的方向及控制过程的执行,从而使程序设计更灵活

变化更多样化。

4.5.1 IF

执行模式：直接与定义模式

格式：IF < 条件表达式> THEN < 活动1 >

IF < 条件表达式> THEN < 活动1 >

ELSE < 活动2 >

功能：第一种形式是根据条件表达式的值是否为真(TRUE)，若为真则执行< 活动1 >，否则为假(FALSE)就什么也不做，接下去执行下一命令行。此命令 THEN 可省略。

第二种形式是根据条件表达式的值是否为真，若为真则做< 活动1 > 内容，否则做< 活动2 > 内容。

说明：< 活动1 >、< 活动2 > 为一系列命令。< 条件表达式 > 可以是由符号>、<、= 联结的两个算术表达式的值而表示其间的大于、小于、等于、小于等于、大于等于或不等于的关系。例如

:X > 10， :X + :Y < =:Z， :N < > 100 等。

下面举两个例子说明 IF 命令的用法。

例1.

```
TO BACKWARDS :NUMBER
IF :NUMBER=0 STOP
PRINT :NUMBER
BACKWARD:NUMBER-1
CTRL-C
END
?BACKWARDS 10
10
9
8
7
6
5
4
3
```

```
2
1
```

过程中第 2 行对:NUMBER 的值进行测试,若等于0 则停止过程执行,否则继续执行下一命令行。

例2. 上例也可改为:

```
TO BACKWARDS :NUMBER
IF : NUMBER =0 THEN STOP ELSE PRINT
:NUMBER
BACKWARDS :NUMBER - 1
END
```

4.5.2 TEST

执行模式:直接与定义模式

格式:TEST < 条件表达式 >

功能:对 < 条件表达式 > 进行测试,若测试为真,即条件成立,则结果为(TRUE),否则为假(FALSE)。

4.5.3 IFTRUE

执行格式:直接与定义模式

格式:IFTRUE < 活动 >

缩写形式:IFT < 活动 >

功能:根据最近一次 TEST 命令的测试结果,若结果为真(TRUE),则执行 IFTRUE 命令中的 < 活动 >。

4.5.4 IFFALSE

执行格式:直接与定义模式

格式:IFFALSE < 活动 >

缩写形式:IFF < 活动 >

功能根据最近一次 TEST 命令的测试结果,若结果为假(FALSE),则执行 IFFALSE 命令中的 < 活动 >。

例:对上例也可改为:

```
TO BACKWARDS :NUMBER
```

```
TEST  :NUMBER =0
IFTRUE  :STOP
IF FALSE PRINT : NUMBER  BACKWARDS  :
NUMBER - 1
END
```

4.5.5 ALLOF

执行模式:直接与定义模式

格式:ALLOF < 条件表达式1 > < 条件表达式2 > ……
< 条件表达式 N >

功能:如果 N 个条件表达式均为真(TRUE),则输出为真(TRUE),如果 N 个中有一个以上的条件表达式为假(FALSE),则输出为假(FALSE)。

例:判断输入的三条边长能不构成一个三角形。

```
TO  TRIANGLE  :A  :B  :C
MAKE  "D  :A+ :B >:C
MAKE  "E  :A + :C >:B
MAKE  "F  :B + :C >:A
TEST  (ALLOF  :D  :E  :F
IFTRUE  PRINT  〔YES〕
IFFALSE  PRINT  〔NO〕
END
```

若执行键入边长为3、4、5 则输出 YES;若键入3、4、1 0 则输出NO。

4.5.6 ANYOF

执行模式:直接与定义模式

格式:ANYOF < 条件表达式1 > < 条件表达式2 >......
< 条件表达式N >

功能:如果N 个条件表达式中有一个以上为真(TRUE),则输出为真(TRUE),如果N 个条件表达式都为假(FALSE),则输出为假(FALSE)。

例: 判断一个三角形是否等边三角形。

```
TO  TRIANGLE  :A  :B  :C
MAKE  "D  :A =:B
MAKE  "E  :B =:C
MAKE  "F  :C =:A
TEST  (ANYOF  :D  :E  :F)
IFTRUE  PRINT  〔YES〕
IFFALSE  PRINT  〔NO〕
END
?TRIANGLE  5  5  7
YES
?TRIAMGLE  3  4  5
NO
```

4.5.7 NOT

执行模式: 直接与定义模式

格式: NOT < 表达式 >

功能: 输出为 < 表达式 > 的反, 即 < 表达式 > 为真, 输出则为假; < 表达式 > 为假, 输出则为真。

例:

```
?PRINT  NOT  5 > 4
FALSE
?PRINT  NOT  4 > 5
TRUE
```

4.5.8 REPEAT

执行模式: 直接与定义模式

格式: REPEAT < 算术表达式 > 〔< 命令表 >〕

功能: 使 < 命令表 > 代表的程序段重复执行, 重复执行的次数为 < 算术表达式 > 的值决定。

说明: 算术表达式的值必须大于等于零, 否则将出现 THERE IS NO PROCEDURE NAMED EPREAT。

例1. ?REPEAT 4 〔FORWARD 50 RIGHT 90〕

使图标重复4次，每次图标前进50步，向右转90度，画出一个边长为50的正方形。

例2. ?REPEAT 4〔REPEAT 4〔FD 50 RT 90〕RT 90〕

画出一个由四个边长为50的正方形组成的一个方字。

4.5.9 STOP

执行模式：直接与定义模式

格式：STOP

功能：停止当前过程的执行，返回到调用当前过程的过程。

说明：STOP不是终止计算机运行，而是使正在被执行的过程终止执行，使计算机反回到刚被终止的调用过程的相继一行。

例：

```
TO AB
ABC
PRINT 123
END
TO ABC
STOP
END
```

这是有两个过程的程序，AB过程中将调用ABC过程执行过程后，可看到

```
?AB
123
```

4.5.10 GO

执行模式：定义模式

格式：GO < 标号>

功能：无条件转移到< 标号> 所指出的命令行上去执行。

说明：这个命令只能用在定义模式中，若用在直接执行模式则会出现：GO SHOULD BE USED ONLY INSIDE A PROCEDURE。所转移到的命令行的标号，必须在过程中存在，否则将出现THERE IS NO LABEL ××:IN LINE GO ××。此外在GO命令之后不能再有其它命令。

例: 设计一个符号函数

$$f(X)=\begin{cases}1 & X>0\\0 & X=0\\-1 & X<0\end{cases}$$

```
TO  SGN  :X
IF  :X > 0  THEN  MAKE  "Y  1  GO  10
IF  :X = 0  THEN  MAKE  "Y  0  GO  10
MAKE  "Y - 1
10: (PRINT  [Y =]  :Y)
END
?SGN  100
Y = 1
?SGN  0
Y = 0
```

GO 命令的< 标号> 可以是一个具体的数字,也可以是一个算术表达式。上例中的第二、三命令行也可以改为:

```
IF  :X > 0  THEN  MAKE  "Y  1  GO  2 * 5
IF  :X = 0  THEN  MAKE  "Y  0  GO  5 * 2
```

最后应注意在所转移到的标号后面一定要用":"作为后缀。

4.5.11 OUTPUT

执行模式:定义模式

格式:OUTPUT < 表达式>

缩写形式: OP < 表达式>

功能:从当前过程中向调用它的过程传送此命令,表达式的结果作为当前过程的结果信息。并把控制权交还给调用它的过程。

例: 已知一个圆的面积,求出它的半径。

```
TO  GETRADIUS  :AREA
OUTPUT  SQRT  :AREA / 3.14159
END
?PRINT  GETRADIUS  10
1.78412
```

在程序中先用 PRINT 命令调用过程 GETRADIUS，在过程执行到 OUTPUT 时，使 SQRT 的结果送回 PRINT 命令，最后打印出结果。

4.5.12 PAUSE

执行模式：定义模式

格式：PAUSE

功能：暂停过程的执行，这时可检查局部的各种状况，以了解错误发生的原因，这在调试程序时是很有用的。执行 PAUSE 后，机器把控制权交给用户。

例：

```
TO  INC  :A5
IF  :A =10  PAUSE
PRINT  :A
INC  :A + 1
END
```

执行后可得到：

```
?INC  1
1
2
3
4
5
6
7
8
9
10
PAUSE,  IN  LINE
IF  :A =10  PAUSE
AT  LEVEL  10  OF  INC
LT 10?
```

4.5.13 CONTINUE

执行模式:直接执行模式

格式:CONTINUE

缩写格式:CO

功能:过程在执行 PAUSE 暂停后,执行 CONTINUE 可使该过程继续执行。

上例 PAUSE 后再键入 CONTINUE 可使过程继续执行。

```
    .
    .
    .
L 1 0?   CONTINUE
1 0
1 1
1 2
1 3
    .
    .
    .
```

4.5.14 TOPLEVEL

执行模式:定义模式

格式:TOPLEVEL

功能:从当前过程中立即返回到直接执行模式。

说明:这个命令与 STOP 不同,它不能返回到调用它的过程上去;另外与 PAUSE 也不同,不能用 CONTINUE 来恢复过程执行。

例:
```
TO   INC   :A
IF   :A = 1 0   TOPLEVEL
PRINT   :A
INC   :A +  1
END
?   INC   6
6
7
```

```
8
9
?■
 ↑
 光标
```

4.5.15 TRACE

执行模式: 定义模式与直接模式

格式: TRACE

功能: 这是一个跟踪执行命令。使 LOGO 在每执行一个过程行前暂停，打印出将执行的过程行，按任意键后，机器执行该行内容，直到过程结束。该语句在调试程序时是非常有效的。

例: 使用跟踪命令执行一个平方表过程。

```
TO  SQUARE  :A  :B
IF  :A =:B  STOP
(PRINT  :A  :A * :A)
A  :A + 1  :B
END
?TRACE
TRACING  ON                (执行 TRACE 后的反应)
?SQUARE  1  1
EXEUTING  1  10
IF  :A =:B  STOP  ■
(PRINT  :A  :A * :A)  ■
1   1
A  :A + 1  :B  ■
EXEUTING  SQUARE  2  10
IF  :A =:B  STOP  ■
(PRINT  :A  :A * :A)  ■
2   4
A  :A + 1  :B  ■
EXEUTING  SQUARE  3  10
```

⋮

EXEUTING SQUARE 10 10

IF :A =:B STOP ■

ENDING SQUARE

注: ■表示该处需按任意键。

4.5.16 NOTRACE

执行模式:直接执行模式与定义模式

格式:NOTRACE

功能:撤消跟踪命令。

例: 上例中若键入

?NOTRACE

TRACING OFF (撤消跟踪)

4.6 变量赋值与输入、输出命令

这一部分将介绍给变量赋值的命令,以及用户与计算机之间交换信息—输入、输出的命令。

4.6.1 MAKE

执行模式:直接与定义模式

格式:MAKE "<变量名> <表达式>

功能:把<表达式>的内容赋给变量。

说明:MAKE 命令可给变量赋于各种内容。

. 把数值赋给变量。

例: ?MAKE "A 123

把数值123赋给变量 A。

. 把一个词赋给变量。

例: ?MAKE "A "WORD

把词"WORD"赋给变量 A。

. 把一个表赋给变量。

例:　?MAKE　"A　[ZHONG　HUA　XUE　XI　JI]

把"ZHONG　HUA　XUE　XI　JI"这个表赋给变量A。

. 把算术或逻辑表达式的值赋给变量。

例1. ?MAKE　"A　3 + 5

把算术表达式3 + 5 的值8 赋给变量 A。

例2. ?MAKE　"A　5 > 3

把逻辑表达式5 > 3 的值(TRUE)赋给变量A。

最后应特别注意:在所被赋值的变量名前一定要用双引号作为前缀,而表达式中的变量名前用":"作为前缀。

4.6.2　PRINT

执行模式:直接与定义模式

格式:PRINT　< 表达式>

缩写形式:PR　< 表达式>

功能:打印表达式的值,并换行。

说明:PRINT 命令可打印多种内容。

. 打印数字。

例:　?PRINT　3.14159

3.14159

. 打印变量或算术表达式的值。

例:　?MAKE　":R　10

?PRINT　:R

10

?PRINT　:R * :R * 3.14159

3.14159

. 打印词、表

例:　?PRINT　"COMPUTER

COMPUTER

?PRINT　[CHINESE　EDUCATION　COMPUTER]

CHINESE　EDUCATION　COMPUTER

若希望在同一行上打印多种形式的内容,则可将 PRINT 命令与

打印内容用圆括号括起来。

例 ?(PRINT 〔THE RESULT IS〕 4 * 4)

THE RESULT IS 16

PRINT 命令也可用来打印空白行。

例: ?PRINT 〔 〕

就可得到一个空行。

最后应注意:所打印的表达式的值中出现的变量必须已赋过值,且在变量的前面用“:”作为前缀。

例: ?PRINT :AB

AB DOESN'T OUTPUT (表示 AB 未赋值)

4.6.3 PRINT1

执行模式:PRINT1 < 表达式 >

功能:打印表达式的值,但打印完后并不换行。

说明:所能打印的内容与 PRINT 命令相同。

例:
```
TO PRINTIT
PRINT1 〔LET、US〕
PRINT 〔LEARN CEN-LOGO〕
END
?PRINTIT
LET US LEARN CEC-LOGO
```

注意:在打印完 PRINT1 的内容后,紧接打印 PRINT 内容,中间不留空格。

4.6.4 REQUEST

执行模式:直接与定义模式

格式:< 命令 > REQUEST

缩写形式:< 命令 > REQUEST

功能:此命令的作用是把键入的内容传送给使用它的那个操作 < 命令 >;这个 < 命令 > 为 MAKE 或 PRINT。所键入的内容显示在屏幕上,并以回车作为输入字符的结束。

说明:REQUEST 所传送的是表形式的数据,即使键入的数据两边

没有用方括号括起来。

例:
```
?MAKE "A REQUEST
CEC LOGO              (键入数据)
?PRINT :A
CEC LOGO
?(PRINT [HELLO I'M]REQUEST)
HELLO I'M WANG WU YI
              ↑
           键入数据
```

如果用户想把键入的数字用在计算之中,则必须用下列语句:

MAKE "NUMBER FIRST REQUEST

这里 FIRST 命令的使用是把键入的数字从表中取出。凡用 REQUEST 把数字键入计算机,并要把这些数字用于计算式中,就必须使用这种类型的语句。经处理后,键入的数值就给了变量 NUMBER

4.6.5 READCHARACTER

执行模式:直接与定义模式

格式:< 命令 > READCHARACTER

缩写形式:< 命令 > RC

功能:此命令的作用是把键入的一个字符或一个数字传给使用它的那个操作< 命令 >,这个< 命令 > 是 MAKE 或 PRINT。对所键入的一个字符并不显示在屏幕上,也无须按回车键。

说明:若键入的一个字符为数字,则输入的是一个数值,可以直接参加数字运算。若键入的一个字符是字母,则表示输入的是一个词,可参加词运算。

例:
```
?MAKE "A RC
■                    (等待键入,若此时键入5)
?PRINT NUMBER? :A
TRUE
?MAKE "A RC
■                    (等待键入,若此时键入 G)
?PRINT WORD? :A
```

TRUE

4.6.6 CLEARTEXT

执行模式:直接与定义模式

格式:CLEARTEXT

功能:清文本屏幕,光标移到屏幕的左上角(0,0)处。

说明:在文本模式下的屏幕上有一个光标,指出当前字符输入/输出的位置,通常情况下逐字右移和下移。一个屏幕共分24行,从0~23;每行为40列,从0~39。

4.6.7 CURSOR

执行模式:直接与定义模式

格式:CURSOR <算术表达式1> <算术表达式2>

功能:把光标移到以<算术表达式1>的值为列,<算术表达式2>的值为行的位置上。

说明:算术表达式1的值必须在0~39范围内,算术表达式2的值必须在0~23范围内,否则将出错。

例: ?CURSOR 10 5

则将光标移到第五行、第五列上。

4.6.8 .DEPOSIT

执行模式:直接与定义模式

格式:.DEPOSIT <算术表达式1>,<算术表达式2>

功能:把<算术表达式2>的值写到以<算术表达式1>的值为地址的内存单元中去。

说明:算术表达式1的值必须在0~65535之间,算术表达式2的值必须在0~255之间。利用这个命令可以将机器语言写入内存中。

例:将一段机器语言数据写到起始地址为770的内存中去。

```
TO SUB :DATA
MAKE "AD 770
10:IF :DATA =[ ]STOP
```

```
 DEPOSIT  :AD  FIRST  :DATA
MADE  "AD  :AD + 1
MAKE  "DATA  (BF  :DATA)
GO  1 0
END
```

执行时应键入过程名及十进制机器语言数据，数据以表的形式输入:

```
    ?
SUB 〔1 7 3  4 8  1 9 2  1 3 6  2 0 8  5  2 0 6  1  3  2 4 0
    9  2 0 2  2 0 8  2 4 5  1 7 4  0  3  7 6  2  3  9 6〕
```

这样就把机器语言装入内存。

4.6.9 .EXAMINE

执行模式: 直接与定义模式

格式: . EXAMINE < 算术表达式>

功能: 取出< 算术表达式> 的值所指的内存单元的值。算术表达式的值应在0 ~6 5 5 3 5 范围内。

例: 若上例已将机器语言装入内存，则键入:

```
?. EXAMINE  7 7 0
RESULT: 1 7 3
```

4.6.10 .CALL

执行模式: 直接与定义模式

格式: . CALL < 算术表达式1 > < 算术表达式2 >

功能: 调用内存中以< 算术表达式1 > 的值为开始地址的机器语言子程序。< 算术表达式2 > 为参数地址。两个表达式的值范围在0 ~6 5 5 3 5 之间。

例: 若前一例已把机器语言子程序装入内存，则将下一个程序键入，再调用此过程就可使 LOGO 奏乐。

```
TO  SOUND  :DATA
IF  :DATA =〔  〕STOP
.DEPOSIT  7 6 8  FIRST  :DATA
```

```
.DEPOSIT  769  FIRST  (BF  :DATA)
.CALL  770  768
SOUND  BF  (BF  :DATA)
END
```

调用过程方法为:

```
?SOUND 〔255 160 228 160 205 160 192 160
  171 160 152 160 140 160 128 160 114 160
  102 160 95 160 84 160 75 160 68 160 62 160〕
```

SOUND 后面是乐曲的数据,以表形式输入,数据中的奇数位数代表间阶,偶数位数代表音长。

4.6.11 OUTDEV

执行模式:直接与定义模式

格式:OUTDEV < 算术表达式 >

功能:根据 < 算术表达式 > 的值选择输出设备。

说明:算术表达式的值实际上为外设接口号,CEC-I 中华学习机有六个槽口(0 ~5),并规定 0 号槽口为显示器或电视机;1 号槽口为打印机;3 号槽口为 BASIC 汉字系统,因此 LOGO 不能使用;6 号槽口为驱动器;在未引入磁盘操作系统时,也不能使用;其它几个用户可自行定义。

例 ?OUTDEV 1 (表示接通打印机)

接着可键入 PRINTOUT 打印出一个过程的清单或进行调用过程,把计算机执行过程的结果打印出来。

例: ?OUTDEV 0

使输出回到屏幕上。

4.6.12 PADDLE

执行模式:直接与定义模式

格式:PADDLE < 算术表达式 >

功能:输出以 < 算术表达式 > 的值为号码的游戏操纵杆的值,该值为操纵杆的角度数,其值在(0 ~255)之间。

说明:LOGO 允许接入4 个游戏操纵杆,其值为0 ~3 号,因此算术

表达式的值也必须在0 ～3 之间。

4.6.13 PADDLEBUTTON

执行模式: 直接与定义模式

格式: PADDLEBUTTON　< 算术表达式>

功能: 输出以< 算术表达式> 的值为号码的游戏按钮的状态。当按钮被按下时，其值为 TRUE，否则为 FALSE。

说明: LOGO 允许接入3 个游戏按钮，其编号为0 ～2，因此算术表达式的值也必须在0 ～2 之间。

4.6.14 CLEARINPUT

执行模式: 直接与定义模式

格式: CLEARINPUT

功能: 清除任何前面已打入的字符。

4.6.15 RC?

执行模式: ?直接与定义模式

格式: RC?

功能: 检测是否有字符输入，若有字符输入但还未被读取，则输出为 TRUE，否则输出为 FALSE。

4.6.16 ;

执行模式: 直接与定义模式

格式: ; < 注释内容>

功能: 这是一个注释命令，可对一个过程进行注释。此命令之后不能跟其它命令。

例:
```
TO SQUARE
  ; THIS PROGRAM IS SQUARE NUMBERS
          .
          .
          .
```

4.7 LOGO 表词的处理命令

CEÇ-LOGO 提供了一些命令用来检查、修改各种表与词，这在处理英语或汉语拼音时是非常有用的。

4.7.1 SENTENCE

执行模式：直接与定义模式

格式：SENTENCE < 表1 > < 表2 >

功能：把 < 表1 > 与 < 表2 > 两个表连接起来。

说明：< 表1 > 与 < 表2 > 可以是具体的一个表，也可以是赋过值的表变量。连接后两表中间自动加一空格。

例1. ?MAKE "A 〔ZHONG HUA〕
?MAKE "B 〔XUE XI JI〕
?PRINT SENTENCE :A :B
ZHONG HUA XUE XI JI

例2. ?PRINT SE 〔ZHONG HUA〕 〔XUE XI JI〕
ZHONG HUA XUE XI JI

如果用圆括号把 SENTENCE 和表括起来，则 SENTENCE 命令可以把多个表连接起来。

例： ?PRINT (SE 〔ZHONG HUA〕 〔XUE XI〕 〔JI〕)
ZHONG HUA XUE XI JI

SENTENCE 可把连接起来的表赋给一个变量。

例： ? MAKE " A SENTENCE 〔 ZHONG HUA〕 〔XUE XI JI〕
?PRINT :A
ZHONG HUA XUE XI JI

4.7.2 WORD

执行模式：直接与定义模式

格式：WORD < 词1 > < 词2 >

功能：把 < 词1 > 与 < 词2 > 两个词连接起来。

说明: < 词1 > 与< 词2 > 可以是具体的词,也要以是赋过值的词变量。

例1 ?PRINT WORD "NO"TIME

NOTIME

把 NO 与 TIME 两个独立的词连接起来了。

例2 ?MAKE "A"NO"TIME

?PRINT :A

NOTIME

4.7.3 FIRST

执行模式: 直接与定义模式

格式: FIRST < 字符串 >

功能: < 字符串 > 可以是一个具体的串或赋过值的变量,另外 < 字符串 > 可表示一个< 表 > 或< 词 >。

若串表示一个< 表 >,执行此命令后,取出< 表 > 中的第一个元素。

若串表示一个< 词 >,执行此命令后,取出< 词 > 中的第一个元素。

例1 ?PRINT FIRST 〔I AM A STUDENT〕

I

例2 ?PRINT FIRST "STUDENT

S

4.7.4 LAST

执行模式: 直接与定义模式

格式: LAST < 字符串 >

功能: 此命令与 FIRST 相仿。但取出一个< 表 > 或一个< 词 > 中的最后一个元素。

例1 ?PRINT LAST 〔I AM A STUDENT〕

STUDENT

例2 ?PRINT LAST "STUDNET

T

4.7.5 BUTFIRST

执行模式: 直接或定义模式

格式: BUTFIRST < 字符串 >

缩写格式: BF < 字符串 >

功能: 与 FIRST 的功能相反,从 < 表 > 或 < 词 > 中取出除第一元素以外的其它元素。

例1 ?PRINT BF 〔I AM A STUDENT

AM A STUDENT

例2 ?PRINT BF "STUDENT

TUDENT

4.7.6 BUTLAST

执行模式: 直接与定义模式

格式: BUTLAST < 字符串 >

缩写形式: BL < 字符串 >

功能: 与 LAST 的功能相反,从 < 表 > 或 < 词 > 中取出除最后一个元素外的其它名元素。

例1 ?PRINT BL 〔I AM A STUDENT

I AM A

例2 ?PRINT BL "STUDENT

STUDEN

以上 FIRST、LAST、BUTFIRST 及 BUTLAST 四个命令本身不能用来检查一个 < 表 > 或 < 词 > 中的各个元素。如果要检查串中各元素,则必须编写一个过程。

例: 检查表中各元素的程序为:

```
TO EXAMINELIST :A
TEST :A =〔 〕
IF TRUE STOP
PRINT FIRST :A
EXAMINELIST BUTFIRST :A
END
```

键入命令执行此过程:

?EXAMINELIST 〔I AM A STUDENT〕

I

AM

A

STUDENT

若要检查词中各元素则只需将第二过程序改为:

TEST :A ="

然后执行此过程可看到:

?EXAMINELIST "STUDENT

S

T

U

D

E

N

T

4.7.7 FPUT

执行模式:直接与定义模式

格式:FPUT < 词> < 表>

功能:将< 词> 插入< 表> 的最前面,成为新< 表> 的第一个元素。< 词> 与< 表> 之间自动插入一空格。

例: ?MAKE "A FPUT "CHINESE〔EDUCATION COMPUTER〕

?PRINT :A CHINESE EDUCATION COMPUTER

4.7.8 LPUT

执行模式:直接与定义模式

格式:LPUT < 词> < 表>

功能:将< 词> 插入< 表> 的最后面,成为新< 表> 的最后一个元素。< 词> 与< 表> 之间自动插入一空格。

例:
```
?MAKE  "A  "COMPUTER〔CHINESE  EDUCATION〕
?PRINT  :A
CHINESE  EDUCATION  COMPUTER
```

4.7.9 LIST

执行模式:直接与定义模式

格式:(LIST < 表达式1 > < 表达式2 >.......< 表达式N >)

功能:以各< 表达式> 为元素组成一个表。< 表达式> 可以是词,表等。

例:
```
?MAKE  "A  〔ABCDE〕
?MAKE  "B  "FGHIJ
?MAKE  "C  12345
?PRINT  (LIST  :A  :B  :C)
〔ABCDE〕  FGHIJ  1 2 3 4 5
```

4.7.10 CHAR

执行模式:直接与定义模式

格式:CHAR < 算术表达式>

功能:以算术表达式的值作为 ASCII 码,并产生该 ASCII 码相对应的字符。

说明:CHAR 以 256 为模,因此 CHAR 65 与 CHAR 320 的结果是一样的。

例:
```
TO  ABC  :ASCII
IF  :ASCII =256  STOP
(PRINT  :ASCII  CHAR  :ASCII)
ABC  :ASCII + 1
END
```

键入执行命令后可看到:

```
?ABC  65
65  A
67  B
```

68　C

.
.
.

4.7.11 ASCII

执行模式：直接与定义模式

格式：ASCII　< 一个字符 >

功能：输出一个字符的 ASCII 码值。

说明：这个命令只能在 ASCII 后面跟一个字符或变量（该变量的值也只能是一个字符），否则将出错。

例1.　?PRINT　ASCII　“A

　65

例2.　?MAKE　“A　“A

　?PRINT　:ASCII　:A

　65

例3.　?MAKE　“A　“ABC

　?PRINT　ASCII　:A

　ASCII　DOESN'T　LIKE　ABC　AS　INPUT

显示出错信息。

4.7.12 LIST?

执行模式：直接与定义模式

格式：LIST?　< 变量 >

　　LIST?　< 字符串 >

功能：如果 LIST?后面的内容是一个表，则输出为真（TRUE），否则为假（FALSE）。

例1.　?MAKE　“A　〔STUDENT〕

　?PRINT　LIST?　:A

　TRUE

例2.　?MAKE　“A　“STUDENT

　?PRINT　LIST?　:A

FALSE

4.7.13 WORD?

执行模式: 直接与定义模式

格式: WORD? < 变量 | 字符串 | 数字 >

功能: 如果 WORD?后面的内容是一个词,则输出为真(TRUE),否则为假(FALSE)。

例1. ?MAKE “A “COMPUTER

?PRINT WORD :A

TRUE

例2. ?PRINT WORD 〔COMPUTER〕

FALSE

4.7.14 NUMBER?

执行模式: 直接与定义模式

格式: NUMBER?〈变量| 字符串| 数字〉

功能: 如果 NUMBER?后面的内容是一个数值或由数字组成的词,则输出为真(TRUE),否则输出为假(FALSE)。

例 1. ?PRINT NUMBER? 1234

TRUE

例 2. ?PRINT NUMBER? ”1234

TRUE

例 3. ?PRINT NUMBER? ”ABC

FALSE

4.8 工作区管理及文件管理命令

用户在使用 LOGO 语言来设计程序时,完全不需要关心程序是如何存放在计算机内存中。一个程序(过程集)是存放在计算机内存的工作区内。在工作区内可以存放若干个过程,以下几个命令是用来管理工作区的。

4.8.1 ERASE

执行模式:直接与定义模式

格式:ERASE 〈过程名〉

缩写形式:ER 〈过程名〉

功能:从工作区消除〈过程名〉所指出的过程。若 ERASE 后面无〈过程名〉,则消除最近键入的一个过程。

例: ?ERASE A

消除过程名为 A 的过程。

4.8.2 ERASE ALL

执行模式:直接与定义模式

格式:ERASE ALL

功能:从工作区消除所有的过程。

4.8.3 ERNAME

执行模式:直接与定义模式

格式:ERNAME"〈变量名〉

功能:删除一个已赋过值的变量。使用时应在变量的前面加上双引号作为前缀。

例:
```
?MAKE"A 3.14
?PRINT :A
3.14
?ERNAME "A
?PRINT :A
THERE IS NO NAME A (变量 A 已删除)
```

4.8.4 PRINTOUT

执行模式:直接与定义模式

格式:PRINTOUT 〈过程名〉

缩写形式:PO 〈过程名〉

功能:显示出工作区内由〈过程名〉所指定的所有过程行。

例:若工作区内有过程:

```
TO  BOX
REAEAT  4 〔FD  100  FT  90〕
END
```

当键入 PRINTOUT BOX,回车后可看到:

```
TO  BOX
REPEAT  4 〔FD  100  RT  90〕
END
```

若 PRINTOUT 后面无〈过程名〉则列出最近键入的过程的全部过程行。

4.8.5 PRINTOUT ALL

执行模式:直接与定义模式

格式:PRINT OUT ALL

功能:列出工作区所有过程的过程行及在直接模式下各变量的赋值情况。

4.8.6 POTS

执行模式:直接与定义模式

格式:POTS

功能:列出在工作区内的所有过程名。

4.8.7 GOODBYE

执行模式:直接与定义模式

格式:GOODBYE

功能:重新启动 CEC-LOGO,原先在工作区内的过程等全部消失。

4.8.8 .NODES

执行模式:直接与定义模式

格式:.NODES

功能:检查 LOGO 中有多少个自由结点可用。

例: ?.NODES

```
RESULT:2288
TO  BOX
REPEAT 4〔FD   100  RT   90〕
END
CTRL-C
? .NODES
RESULT:2249
```

.GCOLL

执行模式:直接与定义模式

格式:.GCOLL

功能:清除废结点,从而提供较多的自由结点。

例:
```
? .NODES
RESULT: 2288
? MSKE "A  "ABCDEFG
? .NODES
RESULT:  2267
? MAKE  "A  "123456
? .NODES
RESULT:  2250
:? .GCOLL
.NODES
RESULT:  2271
```

在工作区内包含了定义过的全部过程,这些过程一直长驻内存,并可根据需要多次调整用它们,除非关机或使用 ERASE 命令消除。一旦关机则在内存中的所有信息将全部消失。

为了使这些过程长久地保存,必须把它们保存在软磁盘上,这样下次开机后,仍能将它们调到计算机内存中,供调用。CEC-LOGO 过程存取磁盘的操作是在 DOS 系统下进行的,因此有关 DOS 的操作方法及新空白磁盘的初始化方法请参阅第 2 章中关于驱动器的使用说明。这里仅介绍如何管理 LOGO 磁盘文件。

4.8.9 SAVE

执行模式:直接与定义模式

格式:SAVE"〈文件名〉

功能:将工作区内所有过程作为一个文件存入磁盘,这个文件的名字为〈文件名〉所指出。

说明:磁盘文件名将包括当前工作区的全部内容,而不只是某个单一过程。文件名的首字符应为大写英文字母,不能是数字,例如:PROGRAM、GAME1、MATH 等都可作为文件名。文件名可以与过程名同名,在使用 SAVE 命令时,在〈文件名〉前面一定要加前缀双引号。

例:当工作区内已有过程,而且磁盘也已初始化,那么把磁盘放入驱动器内,键入

?SAVE "PROGRAM (文件名为 PROGRAM)

待驱动器红灯熄灭后,就把当前工作区的过程以文件名为 QROGRAM 存入磁盘。但工作区的内容仍然驻留在计算机内存中。

4.8.10 SAVEPICT

执行模式:直接与定义模式

格式:SAVEPICT "〈图形名〉

功能:把屏幕上的图形以〈图形名〉为名字存入磁盘。

说明:对〈图形名〉的规定与〈文件名〉相同。

例:SHAPE、PIC 等都可作为图形名。

4.8.11 CATALOG

执行模式:直接与定义模式

格式:CATALOG

功能:将磁盘上的所有文件名列出。

例:?CATALOG

此后计算机立刻列出磁盘上的文件目录。如果你在列目录前,已存入一个工作区过程文件,其文件名为 PROGRAM;另一个为图形文件,其图形名为 PIC。当列出目录后,会发现两个文件名分别为:

PROGRAM.LOG

PIC.PICT

计算机自动在过程文件名的后面增加了一个扩展名 LOG，在图形文件名后面加上扩展名 PICT。

4.8.12 READ

执行模式：直接与定义模式

格式：READ "〈文件名〉

功能：将磁盘上以〈文件名〉为名字的 LOGO 文件装入计算机工作区。

说明：在使用该命令时，不必将文件名后面的扩展名.LOG 键入。

例：若磁盘上有 LOGO 文件 PROGRAM，当键入

?READ "PROGRAM

此时驱动器转动，并且红灯亮，当红灯熄灭时，表示你所需的文件已装入工作区，这时可把磁盘取下。

当用 READ 命令将新的 LOGO 过程载入工作区后，原工作区内的过程就被清除掉。

如果用 READ 命令所指定的文件名在磁盘上找不到时，将得到信息：

< ××× >——FILE NOT FOUND

4.8.13 READPICT

执行模式：直接与定义模式

格式：READPICT "〈图形名〉

功能：将磁盘上以〈图形名〉为名字的 LOG 图形装入计算机的图形显示区。

说明：此命令的规定与 READ 命令相似，所不同的是当图形装入图形区后，原图形被消除。

4.8.14 ERASEFILE 与 ERASEPICT

执行模式：直接与定义模式

格式：ERASEFILE "〈文件名〉

ERASEPICT "〈图形名〉

功能：删除磁盘上的文件或图形。

说明：在使用磁盘保存 LOGO 文件或图形时，有时还需对不要的文件进行删除，那么就可以使用这两个命令。但删除时，应特别小心，因为当文件被删除后，就不能恢复。另外对一些特别有用的文件等可使用 DOS 命令进行“加锁”。凡经“加锁”后的文件在文件名目录表中文件名的最左端的 B 字母前面有一个“*”。

4.8.15 DOS

执行模式：直接与定义模式

格式：DOS 〔<DOS 命令>〕

功能：CEC-LOGO 允许用户直接使用某些 DOS 命令这些命令为：

CATALOG	列出磁盘文件目录
DELETE	删除一个文件
BLOAD	装入二进制数据文件
BRUN	运行二进制程序
BSAVE	将二进制文件存盘
LOCK	将文件加锁
UNLOCK	将加锁文件解锁
RENAME	更改文件名
VERIFY	校验

上列 DOS 命令的作用请参阅 2.7 节。下面介绍几个 DOS 命令应用实例：

若将文件 PROGRAM 加锁，则键入：

?DOS 〔LOCK PROGRAM.LOG〕

注意在使用 DOS 命令时，文件名前面不用加双引号，但文件名后面要加扩展名.LOG，对图形文件则加 PICT 扩展名。

4.9 LOGO 函数命令

CEC-LOGO 中有一些标准函数，用户在设计程序时，可直接使用。在使用时函数的自变量可以是一个数值或赋过值的变量；也可以是一个算术表达式。通常表达式两边用圆括号括起来。这些函数分别

是:

(1) 正弦函数

格式:SIN < 算术表达式>

说明:自变量的单位为度数。

(2) 余弦函数

格式:COS < 算术表达式>

说明:自变量的单位为度数

(3) 反正切函数

格式:ANTN < 算术表达式>

说明:自变量的单位为度数

以上三个为三角函数,对三角函数的另外几种表示法都可由这三种函数来构造。

例:反正弦函数=1 / SIN < 表达式>

正切函数=1 / ATAN < 表达式>

(4) 平方根函数

格式:SQRT < 算术表达式>

说明:算术表达式的值一定要大于等于零。

例:

```
?PRINT SQRT 4
 2
```

(5) 求四舍五入的整数近似值函数

格式:ROUND < 算术表达式>

例:

```
?PRINT ROUND 3.1
 3
?PRINT ROUND 5.6
 6
```

(6) 舍去法取整函数

格式:INTEGER < 算术表达式>

说明:函数的值等于算术表达式值的整数部分,即把小数部分舍去。

例:
```
?PRINT INTEGER 4.99
4
?PRINT INTEGER -6.24
-6
```

(7) RANDOM 随机函数1

格式:RANDOM < 算术表达式>

说明:LOGO 语句中有一个随机数发生器,若需产生随机数则可使用随机函数,这个函数能产生0 ~ INTEGER(算术表达式)- 1 之间的随机数。

例:
```
TO RND
REPEAT 10 〔PRINT RANDOM 10〕
END
```

执行此过程能产生10 个0 ~9 之间的随机数。

```
?RUN
2
5
8
3
0
9
1
8
4
6
```

使用该函数时,表达式的值不能为负数,否则将出现 RANDOM DOESN'T LIKE - ×× AS INPUT 错误。

(8) RANDOMIZE 随机函数2

格式: RANDOMIZE

说明: 这个函数能使 RANDOM 函数产生的随机数更加随机化。使用时应放在 RANDOM 函数之前。

例: REPEAT、10〔RANDOMIZE PR RANDOM 1 0〕

(9) 两数相除之余数函数

格式: REMAIDER < 算术表达式1 > < 算术表达式2 >

说明: 这个函数将能得到 < 算术表达式1 > 除以 < 算术表达式2 > 的余数。

例: ?PRINT REMAIDER 3 3 2

1 (即3 3 / 2 的余数为1)

?PRINT REMAIDER 2 0 1 0

0

该函数本是对整数而言,若表达式的值有小数,则先对算术表达式的值进行四舍五入,然后再求两数的余数。

例: ?PRINT 3.5 2

0

?PRINT 2.5 2

1

(1 0) 两数相除之整商函数

格式: QUOTIENT < 算术表达式1 > < 算术表达式2 >

说明: 这个函数能得到 < 算术表达式1 > 除以 < 算术表达式2 > 的商的整数部分。

例: ?PRINT QUOTIENT 3 3 2

1 1

该函数本来也是针对整数而言的,若两个表达式的值中含有小数时,则先对它们进行四舍五入,然后再求得它们商的整数部分。

例: ?PRINT 3 · 5 2

2

?PRINT 4 2.1

2

4.10 直接执行模式的编辑命令

以下这些命令均只能用于直接执行模式,它们对调试程序起很大作用。操作这些命令时,必须将 CTRL 键按下,同时按下另一个指定键,注意必须两个键同时按下才行。

(1) CTRL-F

说明:这个命令可将屏幕置为全屏幕作图方式。

(2) CTRL-S

说明:这个命令可将屏幕置为图形、文本混合屏幕模式。在屏幕上方为图形区,下方留出四行作为文本显示区。

(3) CTRL-T

说明:这个命令将显示屏幕设置为全屏幕文本显示模式。

(4) CTRL-G

说明:当一个过程正在执行过程中,若需中断执行,并返回直接执行模式时,可按 CTRL-G 进行强行中断,但这样处理后,程序不能继续执行了。

(5) CTRL-W

说明:这个命令能停止一个过程的执行,当按任意键后,可使过程恢复执行。

(6) CTRL-Z

说明:这个命令与 PAUSE 命令相同,都可暂停过程执行,所不同的是该命令只能用于直接执行模式中。一个正在执行的过程,当用户按下 CTRL-Z 命令后,会出现:

```
        ⋮
PAUSE,IN   LINE
```

××××××××　　　　　　（暂停所在的过程行）

AT LEVEL ×× OF < 文件名 >

LXX?

此后当键入 CONTINUE 命令后，过程将恢复执行。

(7) CTRL-P

说明：这个命令将能显示出最近键入的一个命令行，光标停留在这个命令行的末尾。

例: ?PRINT "COMPUTER

COMPUTER

?CTRL-P

?PRINT "COMPUTER 

光标

第五章
监控系统和小汇编系统的使用

监控程序是用来管理 CEC－Ⅰ中华学习机系统的重要程序，它被固化在机器 ROM 中的高地址区域＄F800～＄FFFF，占用了 2K 的存储空间。由于中华学习机内存容量的限制，对监控程序要求占用存储空间少且执行速度快，因此监控程序在设计中用机器语言编写，使用了许多编程技巧。

本章将讲述监控系统的特点、用途以及怎样将它同所写的 BASIC 程序配合使用。监控程序是一个功能相当强的程序，其中有许多可被其它程序调用的子程序，用于完成一些机器的基本功能操作，如打印字符，划线等。我们还可以通过键盘命令来使用监控系统，借此产生各种图象形状表，检查存储器以避免硬件出现故障。总之，对监控系统的深入了解，将有助于对 CEC－Ⅰ中华学习机系统软件及应用软件的开发。

在描述了监控系统的使用之后，本章还要说明如何使用小汇编系统，但并不讲解如何使用汇编语言编写程序。要了解这方面的问题，请查找附录中所介绍的有关参考书。在本章中只是说明如何应用小汇编程序系统去设计，测试过程调整机器语言程序。

5.1 监控程序的主要功能

(1)管理 64K 存储器

监控程序具有检查、修改、比较、传送存储器内各种信息的能力，能把大量数据和程序放到存储器中或从存储器中输出，同时还管理系统中的各种工作区，缓存区和特殊向量单元。

(2)管理外部设备

中华学习机内连的Ⅰ／O 设备，如键盘，扬声器、盒式磁带机等，完

全由监控程序来进行管理。对于那些外接的I／O设备,如软盘驱动器,打印机等尽管在连接它们的接口卡上配备有自己的驱动程序,但其驱动程序的启动也是靠监控程序来完成的。

⑶调试程序、简单计算

监控程序为用户调试程序提供了监督功能,用户可以在程序中设置一系列断点,程序运行到这些地方时,监控程序将立即作出反应:中断程序的运行,保护好各种必要的信息,显示断点位置及各种状态,并允许用户从这一点继续执行程序。另外,一些简单的数值计算工作,可直接作为监控命令来实现

⑷为用户提供丰富的子程序

监控程序担当着对系统进行管理的重任,本身包括了各种最基本的系统管理子程序,如完成字符、数字、命令的输入输出的功能,实现延时,响铃功能以及图形处理的功能。这些子程序都是对用户开放的。用户利用这些子程序编制程序,不仅具有很强的功能,而且存储空间的占用也大为减少。

⑸系统初始化

当机器加电时,监控程序将对系统进行初始化,使机器进入工作状态

5.2 进入监控程序

中华学习机监控程序的起始单元是$FF69(十进制的65385或－151),调用此单元就可以进入监控程序。为了进入监控程序,只需在BASIC状态下键入下列命令:

CALL－151

当进入监控程序后,监控程序提示符*(星号)会立即出现在屏幕的最左端,其右侧有一个闪烁的光标。监控程序由此光标开始接收一个标准的输入行。

5.3 退出监控系统

有几种方法可以使CEC－I退出监控系统,而进入BASIC语言系统。

若要保护在内存中的BASIC程序语句和变量,则按下CTRL－C键然后再按回车键就可以退出监控系统,而出现BASIC提示符。这时,如果内存

中有 BASIC 程序的话，就可把变量值和程序清单打印出来。

如果希望在退出监控系统的同时，并清除目前存储器中的程序和变量，则可按下 CTRL- B 键和回车键。这样，在进入 BASIC 语言系统之后，把原来存储器中的任何程序语句及变量都清除了。

5.4 监控程序命令

监控程序的命令行中一般包含三种内容：地址，数据和命令字。其中地址和数据分别用 4 位和 2 位 16 进制数表示。如果监控命令中的 16 进制数的地址少于 4 位，监控程序就在它的前面加 0；若多于 4 位，则只取后 4 位的 16 进制数。监控程序能识别 22 个不同的命令字符，其中包括标点符号、大写字母及控制字符。任何一个监控命令均需在键盘上输入命令之后，再按 RETURN 键才能执行。监控命令执行时显示的内容均采用1 6 进制数表示。

监控命令行的长度受到系统键盘缓冲区大小的限制，它不得超过 254 个字符；否则，监控系统将从本行跳出，并忽略该行所敲入的数据及命令。

在表 5.1 中列出了全部监控命令字符。请注意，输入行中的控制字符并不显示在屏幕上。

表 5.1 监控命令字符

标点符号	+ 『 : . < CR SPC
大写字母	M V L N I G W R S T
控制符	CTRL-B CTRL-C CTRL-E CTRL-P CTRL-K CTRL-Y

其中 CR 表示回车字符，SPC 表示空格字符。控制字符的表示方式意味着：“先按下 CTRL 键，保持按下的同时再按后面的字符键。”

5.4.1 显示存储器的内容

格式 1: 〈地址〉

功能: 显示该地址单元的内容。

说明: 当屏幕上出现了监控系统的提示符后，想要检查某个存储单元的内容，则打入该单元的 16 进制地址，并按下回车键。

例:要检查＄ FF69 单元的内容,则打入:

* FF69

监控系统就把该单元的内容显示出来:

FF69- A9

当送入一个地址后,监控系统就将它保留起来,以备作地址指针。因此,如果输入 FF69,监控系统就记住了这个单元,直至改变了这个地址以前都利用它作为以后进行存储器检查的起始地址。要改变监控系统所设的地址指针,只须直接打入一个新的地址即可。

格式 2: 〈地址 1〉.〈地址2〉

功能: 显示从地址 1 到地址 2 之间所有的存储单元的内容。显示时,当最低位地址为 0 或 8 时,将换行显示。若指定的首址不大于末址,则只显示首址单元的内容。

例:打入:

* F800.F83F

监控系统就按顺序显示出 F800 到 F83F 这段存储区的内容:

```
F800 —4A   08   20   47   F8   28   A9   0F
F808 —90   02   69   E0   85   2E   B1   26
F810 —45   30   25   2E   51   26   91   26
F818 —60   20   00   F8   C4   2C   B0   11
F820 —C8   20   0E   F8   90   F6   69   01
F828 —48   20   00   F8   68   C5   2D   90
F830 —F5   60   A0   2F   D0   02   A0   27
F838 —84   2D   A0   2F   A9   00   85   30
```

利用这种显示方式检查大片存储区时,即使超过了屏幕的显示范围也可以。在这种情况下,数据将不断往上移动,最上部分的内容移出屏幕之外,后面的数据继续从底部显示出来。这时我们可以按下 CTRL- S 键来暂停屏幕上的显示,使之有机会来观察屏幕上所显示的内容。需要重新启动屏幕显示时,可按下空格键。

格式 3: .〈地址 2〉

功能: 显示从现行的存储器地址到地址 2 之间的内容。若地址 2 不大于现行的地址,则仅显示现行地址的内容。

例如：在上述例子中已检查了地址 F800 ~ F83F 这个存储区的内容，则指针已自动指向 F840 单元。因此，可以利用指针 F840 作为起始地址，继续检查 F840 到 F880 这段在储区的内容。于是送入命令：

＊.F880

从而得到下列输出：

```
F840 —20   28   F8   88   10   F6   60   48
F848 —4A   29   03   09   04   85   27   68
F850 —29   18   90   02   69   7F   85   26
F858 —0A   0A   05   26   85   26   60   A5
F860 —30   18   69   03   29   0F   85   30
F868 —0A   0A   0A   0A   05   30   85   30
F870 —60   4A   08   20   47   F8   B1   26
F878 —28   90   04   4A   4A   4A   4A   29
F880 —0F
```

格式 4:　(只按一个回车键)

说明：　显示上次最后访问的下一单元地址到最低地址为 7 或 F 单元的内容。

例如：当我们检查了 FF69 单元的内容以后，还想继续检查下一个更高的存储单元。此时，只要按下回车键，监控系统将相继地显示后面存储单元的内容，如下所示：

```
＊FF69
FF69- A9
＊(按下回车键)
  AA  85  33  20  67  FD
```

5.4.2　改变存储器的内容

格式 1:　〈地址〉:〈数据〉 〈数据〉· · · · · ·

功能：　从地址单元开始顺次将所列数据送入存储器。数据之间加空格，该命令可以把机器语言程序及所需的数据块送入存储器。

说明：　监控系统可以一次修改一个以上存储单元的内容，即修改一个存储区的内容。所修改的存储区地址必须是按顺序排列的(由小到

大)。使用这条命令时,首先要设置地址指针,接着用冒号(:)来告诉监控系统,这是修改存储器内容的命令,最后,按存储单元的顺序,依次输入要送的数据。每一个存储单元所输入的 16 进制数据,必须用空格分开。

例如:将数据 00 ~ 07 按顺序输入到地址为 1200 ~ 1207 的各存储单元中,则输入:

*1200:00　01　02　03　04　05　06　07

格式 2:　:〈数据〉〈数据〉·······

功能:　从上次用来存放数据的地址之后开始,将数据顺次送入存储单元。

例如:在上面的例子中,若我们按回车后,还想修改地址 1207 以后存储单元的内容,这时只需键入:

*:08　09　0A

于是地址 1208 ~ 120A 的内容就被修改成 08,09,0A。

5.4.3 传送存储器中的内容(MOVE)

格式:〈地址 1〉<〈地址 2〉.〈地址 3〉M

功能:　将地址 2 至地址 3 区域内的数据复制到从地址 1 开始的存储单元中去。M 命令只是把源区的内容拷贝到目的区,源区的内容一般并不破坏。

例:将地址 2000 ~ 2100 单元的内容复制到 1200 ~ 1300 单元中,可键入:

*1200 < 2000.2100M

在使用该命令时,如果源区的末地址小于目的区的首地址,那么监控系统将只复制源区首地址中的一个字节数据到目的区的首地址中去,然后就结束M 命令。

M 命令允许源区与目的区地址全部或部分重叠。利用这一性质,我们可以填写存储器的内容。也就是说把一个或多个字节重复地拷贝到连续的存储单元中。

例:给首地址为 1D00,末地址为 1DFF 这段存储区填入 0。其填入过程如下:

① 将 0 放入存储区的第一个字节内:

* 1D00:00

② 打入命令:

* 1D01 <1D00.1DFFM

执行这条命令后,就用 0 填满了 1D01 ~ 1DFF 这段存储区,或者更确切地说,是用 1D00 单元的内容填满了这个存储区。

5.4.4 检查存储器的内容(VERIFY)

格式: 〈地址 1〉<〈地址 2〉.〈地址 3〉V

功能: 检查从地址 2 到地址 3 之间的数据块与从地址 1 开始的长度相等的数据块是否完全相同。

说明: 这个监控命令用来对两个存储区的内容进行对比,指出第一存储区和第二存储区之间有何不同,对那些不相同的单元将显示单元地址和两个不同的内容。执行该命令并不破坏两个存储区的内容。

例:

* 32D0 <0.CV

执行该命令,监控系统要比较的存储区的内容放在首地址为 32D0 的一串单元中,被比较的相对应的存储区的内容放在首地址为 0 的一串单元中。监控系统首先将 32D0 单元的内容与 0 单元的内容相比较,然后,依照顺序,继续两两对应相比较,一直到地址 32DC 和对应的 000C 比较为止。

5.4.5 将存储器的内容存入盒式磁带(WRITE)

格式: 〈地址 1〉.〈地址 2〉W

功能: 将规定的存储器区域(即〈地址 1〉至〈地址 2〉)内的数据写到盒式磁带上。

例:下面这条命令:

* 2200.2FFFW

意即告诉监控系统,写在磁带上的这段存储区的内容,它的起始地址是 2200,结束地址是 2FFF。

存储器写命令不能检查它送到盒式磁带上的数据,也不能检查连接磁带录音机现行的实际运行状况。所以,在使用 W 命令时,必须确保所用的磁带不出现问题。在输入行按回车键之前,须先按下磁带机的

录音键，过数秒钟以后再按回车，以保证能得到正确的记录。

当按下回车键以后，光标消失，表示机器正在写带。监控程序首先在磁带上写上大约 10 秒钟的“引导”信号，然后才写数据信号。当传送完最后一个数据之后，监控系统送一个“校验和”字节来检查读入的数据是否正确。

当写命令完成以后，机器会发出“嘟”的一声，同时，监控系统提示符将出现在屏幕上。

5.4.6 从盒式磁带中读取数据(READ)

格式: 〈地址 1〉.〈地址 2〉R

功能: 将盒式磁带上的数据读入所规定的存储器地址区域内。

例: 下面这条命令:

* 2000.20FFR

即意味着，从盒式磁带上读出的数据输入到这样一段存储区，其起始地址为 2000，结束地址为 20FF。

执行读命令时，计算机首先要读出磁带上数据段前面的引导信号，在未读出这个引导信号之前，监控系统一直将计算机锁定。因此在给出读命令按回车键之前，首先要把磁带转到准备读取的数据段的引导信号之前(使用录音机的扬声器来听磁带的声音时，可以听出: 引导信号是始终不变的中频音调嗡声，而实际的数据声音象随机的噪声或无线电干扰声)，待走带到引导信号处时，立即按回车键。另外，在使用读命令之前，必须事先调节好磁带机的发音音量。

在使用读命令时，要求读取磁带数据的长度(即〈地址 2〉- 〈地址 1〉)与写带的数据长度相等，写入多少，就应读回多少，否则机器会显示读带出错信息，或出现“死机”现象。但是读带命令可以把记录在磁带上数据放在任一存储区域中，不一定要放在写带命令指定的区域中。

监控程序读完磁带上的数据后，它自己生成一个“校验和”，再与从磁带上读出的“校验和”比较。若两者相等，则表示读带正确，显示出监控系统提示符; 否则显示“ERR”出错信息，表示有错误，读入的数据不能使用。

5.4.7 设定屏幕显示方式(INVERSE / NORMAL)

格式 1:　I

功能:　置屏幕显示方式为反相显示,这时在屏幕上显示的字符为白底黑字。

格式 2:　N

功能:　置屏幕显示方式为正常显示,在屏幕上显示的字符为白底黑字。

5.4.8　反汇编命令(LIST)

格式 1:　〈地址〉L

功能:　将指定地址开始的 20 条机器语言指令翻译成 6502 助记符,并在屏幕上显示出来。

格式 2:　L

功能:　将现行地址开始的 20 条指令翻译成 6502 助记符,并显示在屏幕上。

5.4.9　执行程序命令(GO)

格式:　〈地址〉G

功能:　将机器的控制权转移到所指定的地址去执行机器语言程序,并且在执行完一条 BRK 指令后中断。

说明:　监控程序在执行 G 命令之前,首先要恢复上次中断时所保存起来的各寄存器(A- P),这些内容保存在地址 $ 45 ~ $ 48 中。如果命令中省略了转移地址(只给出 G),则把上次中断时保存起来的 PC 值作为执行地址。

例:下面的命令执行后将退出监控系统,并把控制权交给 DOS 系统:

　　* 3D0G

5.4.10　显示和修改 CPU 寄存器的内容(EXAMINE)

格式:CTRL- E

功能:　依次显示 6502 累加器 A,变址寄存器 X、Y,状态寄存器 P,以及堆栈指示器 S 的当前内容。

说明:修改 CPU 中寄存器的内容与修改存储器中的内容,其过程有所不同。这是因为,寄存器没有地址。要修改寄存器中的内容,首先必

须使用 CTRL- E 命令来检查它们,接着输入一个冒号。然后就可用空格为界,键入 1 到 5 对 16 进制数,它们将按上述寄存器的先后顺序,分别取代原寄存器的内容。

在修改某个寄存器中的内容时,必须先输入排列在这个寄存器之前的所有其它寄存器值,而把指定修改内容的那个寄存器作为最后一个寄存器,使它放入要修改的值。排列在被修改的寄存器之后的其它寄存器,可以不赋值,监控系统将保留原值。

例:假如我们要修改变址寄存器 Y 的内容,而其它各寄存器的内容保持不变,其方法是这样的:

① 首先用 CTRL- E 检查各寄存器中的内容:

*CTRL- E(然后按回车键)

A =CD　　X =B1　　Y =C3　　P =B5　　S =F0

② 若要将数值 8A 送至变址寄存器 Y 中,而使其它各寄存器中的内容不变,可输入:

*:CD　B1　8A(按回车键)

③ 接下来我们可以再次用 CTRL- E 检查已修改后的各寄存器的内容,证实一下所修改的值是否正确:

*CTRL- E(回车)

A =CD　　X =B1　　Y =8A　　P =B5　　S =F0

5.4.11 选择输出设备(PRINT)

格式: 〈槽号〉CTRL- P

功能: 将输出控制权转给槽号所指定的连接槽上的接口卡。槽号只能取值为 1,2,4,5,7。若槽号为 0,则返回以显示器为输出设备的状态。若槽号为 6,则转入磁盘驱动程序执行冷启动。

说明:使用 CTRL- P 命令时,一定要在所选用的槽号上,先插入符合要求的接口插件板。槽号可通过主机板右侧的连接开关进行设置,一般机器出厂时设置为 1 号槽。如果在规定的那个插槽上没有插入相应的接口板或与接口板连结的输出设备没有准备好,那么,机器就会被锁定住。而要从这种状态得到恢复的唯一办法,就是按下 CTRL- RESET 键。

例:将打印机指定为输出设备的命令为:

＊CTRL－ P

注:CTRL－ P 命令只有在＄ C006 软开关置位的情况下,方能有效地使用。若槽号为 3 时,则还需＄ C00B 软开关置位。在系统启动后,这两个软开关均已自动置位。本命令不能在中文状态下使用。

5.4.12 选择输入设备(KEYBOARD)

格式: 〈槽号〉CTRL－ K

功能: 当槽号为 1,2,4,5,7 时,将输入控制权转给槽号所指定的连接槽口上的接口卡。当槽号为 0 时,则返回以键盘为输入设备的状态。该命令的使用方法与 CTRL－ P 相同。

5.4.13 十六进制加、减法运算

格式: 〈数据 1〉±〈数据 2〉

功能: 实现以 256 为模(8 位数)的十六进制加减运算,并将结果显示在屏幕上。

说明: 进行加法运算时,若结果大于 FF,监控系统会自动舍去最高有效位,显示出结果的低八位数字。

例:＊7F＋8A

=09

进行减法时,如果结果小于零,监控系统就用补码的形式将结果显示出来。

例:＊0A－ 2D

=DD

5.4.14 单步执行(STEP)

格式: 〈地址〉S

功能: 执行指定地址处的一条机器指令,并将该指令的反汇编助记符显示出来。指令执行完后,各寄存器的值也同时在屏幕上给出。如若想继续单步执行下一条机器指令,可再敲入一个 S(以回车键结束)。此过程可以一直进行下去,直到不再需要执行程序为止。

说明: S 命令提供了一种单步检查的功能,在程序很小的情况下,可以一步一步来执行它,并在程序执行后,检查每一条机器语言指令执行

后的结果，从而查出错误来。

5.4.15 跟踪执行(TRACK)

格式： < 地址>T

功能： 从指定地址处开始执行机器语言指令，每条指令执行完后，其指令的反汇编助记符和各寄存器的值被显示出来。

说明： 当需要了解一个较长程序的执行情况时，往往由于程序太长，使得无法一步一步地去执行每一条指令并检测它。T 命令可以让你观察到每条指令的执行情况，而只有有必要时才打断程序的执行。T 命令的输出与 S 命令类似，但它不象 S 命令那样，每检查一条指令都要输入一次 S 命令。在需要停止 T 命令时，可以按下 CTRL- RESET 键，也可在程序中插入一条 BRK 指令。当监控系统遇到这条 BRK 指令时，就会中断程序的执行。

顺便指出，T 命令会使程序的执行速度变得缓慢得多。因此，如果需要测试一个较长的程序，而又想用 T 命令来帮助的话，就应该慎重地考虑，以免在程序执行过程中，浪费太多的时间。

5.5 监控命令的灵活使用

5.5.1 使用多重命令

CEC- I 监控系统允许在一行中安排若干条监控命令，各命令之间以空格分开，这些命令会被监控系统顺序执行。

多重命令一行中的字符总数不能多于 256 个，当字符数达到 248 个以后，每输入一个字符，监控系统都给以响铃警告。当输入字符超过 256 个时，监控系统会给出“/”号，表示此行作废，并回车换行，显示新的提示符。

在多重命令中若使用了修改存储器命令，则在该命令之后，应写一个单一字母的命令(通常用 N)，用于把随后的命令隔开。

下面的例子是先显示一个地址范围的内容，然后修改一些单元，并再次显示这些被修改单元的内容。

```
* 300.307   300:18   69   01   N   300.302
0300 — 00   00   00   00   00   00   00   00
0300 — 18   69   01
```

5.5.2 复制某种格式的数据到一个存储区

使用监控系统的 M 命令,可以把某种格式的数据复制到存储器中的某一区域,具体做法如下。

首先把数据模型存入该区域的首地址。接着使用如下格式的 M 命令:

〈源首地址+ n〉<〈源首地址〉.〈源末地址 − n〉M

其中 n 是数据模型的字节数。

例: 在存储区域 300 ~ 32F 填入数据 11,22,33,具体作法如下:

```
 *300:11   22   33
 *303 <300.32DM
 *300.32F
0300 — 11    22    33    11    22    33    11    22
0308 — 33    11    22    33    11    22    33    11
0310 — 22    33    11    22    33    11    22    33
0318 — 11    22    33    11    22    33    11    22
0320 — 33    11    22    33    11    22    33    11
0328 — 22    33    11    22    33    11    22    33
```

5.5.3 建立可无限重复执行的命令行

使用一个命令可以反复地执行一个或几个命令。此命令行的重复执行开头必须是一个单字命令(通常用 N),命令行的末尾必须是:"34:n"。在此 n 是一个 16 进制数,它指定命令行中开始重复的位置。n =0 表示从第一个字开始重复。为使循环正常进行,第 n 个字符的后面必须有一个空格。

例如,以下命令将重复显示 300 ~ 302 这两个单元的内容:

 *N 300 302 34:0(回车)

要结束该命令行的执行,唯一的办法是用 CTRL− RESET 键。

5.6 在程序中使用监控子程序

在某些情况下,BASIC 语言系统并不能圆满地完成程序中所要求的全部功能,而 CEC− I 监控系统为用户提供了大量的功能十分丰富

的子程序。了解这些子程序的功能并学会正确地使用，会使我们自己编制程序时感到十分方便，并使编制的程序具有较强的功能。

在本书的附录 10 中列出了一些常用子程序的功能，入口地址，使用条件及执行结果。如果在你的程序中决定采用监控子程序，那么首先要知道所调用的子程序是否需要由你的程序事先设置好相应的存储单元和寄存器的初值，以及子程序的执行结果是否需要传回到程序中。

如果不需要与子程序之间传送参量，那么就可以直接用 CALL 语句调用执行子程序。例如 CALL− 936，所调用的子程序的功能是把文本显示的屏幕清除，并把光标移到屏幕的左上角。为了使这条 CALL 语句具有更强的说服力，使程序更易阅读和修改，可在程序的起始处设置一个富有含义的变量 CLSCREEN（清除屏幕的意思），如下所示:

```
10   CLSCREEN=− 936
```

然后，在这个程序的后面再使用它。

```
1510  CALL   CLSCREEN
```

在用 BASIC 语言设计程序时，如果必须与子程序之间传送参量，那么就必须用另外的机器语言程序来与 BASIC 程序相互连接，这时总是使用 USR 函数，而不使用 CALL 语句。在地址 9D ～ A3 的存储区中，存储了用 USR 函数传送的参量值，同时，也可以使用这个区域将参量传送回 BASIC 程序中。若要使用 USR 函数使得机器语言子程序开始执行，则必须使用 POKE 语句放一条 JMP 指令到 0A ～ 0C 单元。因为在这些单元内必须包含一个可供 USR 函数调用的 JMP 指令，才能由它把控制权转到机器语言子程序的开始处，以执行机器语言子程序。

5.7　CEC− Ⅰ小汇编系统

小汇编系统是中华学习机固化在 ROM 中的汇编工具软件，其功能是将 6502 汇编助记符翻译成 6502 机器码指令。用小汇编来编写机器语言程序时，可以避免了人工翻译的麻烦，用户可以直接用 6502 指令的助记符编写程序，小汇编系统会自动将这些助记符汇编成机器语言送到 CEC− Ⅰ内存中。但是用小汇编系统时，6502 指令的操作数必须使用实际的存储器地址而不能用标号，它的主要问题是不能象拥有带文

本编辑的汇编器那样,可以随意插入或删除要修改的指令。因此小汇编系统常被用来编写一些简单的机器语言程序。

本章节将详细描述小汇编系统及其应用,但并不讲述汇编语言程序的设计方法,也不对 CEC- I 所使用的 6502 汇编语言指令系统作全面的说明。所以,如果对汇编语言、操作数,助记符等基本概念还搞不清楚的话,请不要急于阅读本章节,建议首先学习汇编语言程序的设计方法和 6502 指令系统,然后再来阅读本节。

5.7.1 小汇编系统的进入及退出

小汇编系统的入口地址是 $ D350,因此在监控状态下敲入下列命令行,即可退出监控系统而进入小汇编系统:

* D350G

当进入小汇编系统后,机器会发出“嘟”的响声,同时屏幕会显示小汇编系统的提示符“!”。

当需要退出小汇编系统而进入监控系统时,只需敲入下列命令行:

! $ D360G

5.7.2 在小汇编系统中使用监控命令

在小汇编系统中,任何时候都可以执行监控系统的命令。其方法是在小汇编系统的提示符“!”后面,立即输入符号“ $ ”,紧跟着输入一个监控系统命令。其格式如下:

! $〈监控系统命令〉

下面举一个例子,说明在小汇编系统内如何检查存储器的内容。

! $ 1CFF

1CFF- F6

!

小汇编系统的这个功能节省了在小汇编与监控系统之间来回转换的时间。事实上,当要退出小汇编系统时,也必须使用这个功能。

5.7.3 小汇编系统的使用

在小汇编状态下,可按下列格式进行工作:

〈地址〉:〈汇编助记符〉

每敲入一行上述格式的命令，小汇编程序便汇编一条6502汇编指令，即将对应的机器指令存入冒号(:)前面的地址。若指令为多字节指令，则存入以该地址为首地址的连续几个存储单元。

在输入指令行时，助记符和操作数之间必须有一个空格来间隔开。当输入指令行之后(按回车键)，小汇编系统将会把输入的指令显示在屏幕上，显示格式如下：

地址—机器码　助记符　操作数

例： !300:LDA　# $ 01

0300- A9　01　LDA　# $ 01

在输入完第一条指令之后，输入下一条指令时，可以省略地址的键入，小汇编程序会自动计算下一个地址，除非你按照第一条指令输入的方式重新设定另外一条指令的地址值。

只要设定了最初指令的地址之后，就可以一行一行地输入一系列的汇编语言指令，这时只要在输入指令之前留一个空格位置即可。

例：　!　ASL(注意:!与ASL之间留有一个空格)

0302- 0A　ASL

如果在指令之前没有空格，那么这条指令就不会被接受。每输入一条汇编语言指令时，小汇编系统能自动地进行查错。当它发现输入过程中的错误时，就发出"嘟"的一声来指示错误，并且用一个插入符(∧)显示在指令中首先出错的那个字符下面。这时计数器的内容不增加。由于前次送入的字符仍保持不变，所以可再送入正确的指令。

在输入完一个程序之后，还必须检查输入工作是否完全正确。最好的检查是用监控系统的L命令，用汇编语言形式将存储器中的程序清单显示出来。完成这一工作后，便可使用监控命令进一步进行下面的一些工作。

也许你想要把这个程序安全地保护起来，那么，就用监控系统的W命令，把这个程序存入盒式磁带上，或者用BASIC语言中的BSAVE命令，把程序存入磁盘上。运行这个程序时，必须使用监控系统中的G命令，或者BASIC语言中的CALL命令转入程序的入口地址去执行程序。

注意，中华学习机小汇编程序只宜在西文状态下使用，而不能在中文状态下使用，这是因为中西文状态在存储器空间上存在着冲突。

第六章 程序中磁盘文件的使用

我们知道计算机内存可以和磁盘交换信息，这些信息在磁盘上的集合，称为文件。

一个 BASIC 程序，在内存中实际上是一批信息的集合，通过 SAVE 命令将其存入磁盘，变成磁盘上的一组信息的集合。这就是 BASIC 程序文件。一个程序在运行时，会输出一些数据。如果把这些数据输出到磁盘上，它们在磁盘上的集合，称为文本文件。文本文件中的信息一样可以读入程序中使用。

对于文本文件的使用，不仅相当于扩大了内存的容量，使一些数据量大的得以运行，而且提供了长久保存数据的手段。

文本文件按其存取方式可以分成顺序文件和随机文件。这两种文件在目录中的类型标志均为"T"，但在使用方法和存储格式上却有所不同。

6.1 在程序中使用 DOS 命令

在 BASIC 程序中，DOS 命令均可作为一条特殊语句被调用。它与一般 BASIC 语句不同的是，DOS 命令必须通过一条特殊的 PRINT 语句来执行。这条 PRINT 语句将 DOS 命令作为一输出行进行输出，而在这输出行的第一个字符必须是控制符 CTRL- D(ASCII 码 4)。其格式如下：

PRINT "CTRL- D DOS 命令"

或 PRINT CHR $ (4);"D0S 命令"

当 BASIC 解释程序执行 PRINT 语句时，遇到第一个输出字符为 CTRL- D 后，则作为 DOS 命令去处理，而不会将其后面的 DOS 命令字符串作为一般的输出字符显示在屏幕上或输出设备上(当然，如果系统中没有启动 DOS 操作系统，BASIC 程序是不会对 CTRL- D 字

符作特殊处理的)。

可以简单地定义一个变量来代表 CTRL- D 或 CHR $ (4)。变量名可以任意选用,在程序中,一般用 D $ 代表。例如:

D $ ="CTRL- D"或 D $ =CHR $ (4)

下面的程序只要软盘中存的文件不超过 18 个,可不必打入任何键,看到 10 次目录显示。

```
10  D $ =CHR $ (4)
20  FOR I=1 TO 10
30  PRINT D $;"CATALOG"
40  NEXT I
50  END
```

在使用 BASIC 语句调用 DOS 命令时应注意:在一条 PRINT 语句输出行中,只能含有一条 DOS 命令,必须以 CTRL- D 字符打头,以回车字符表示 DOS 命令的结束。

6.2 顺序文本文件的管理

6.2.1 顺序文件的存放格式

文件中的数据,都是以每个字符的 ASCII 码的形式存放的,每个字符的 ASCII 码占一个字节的位置。通常我们把由若干个字符组成的字符串称为字段,由若干个字段组成一个记录,由若干个纪录组成一个文件。

顺序文件中的记录在软盘上是连续存放的,每个记录以回车符作为终结标志。字段的终结符可以是回车符也可以是逗号","。例如,我们往磁盘上存放如下信息: 7 AT ONE BLOW,OP

其中每个字符串后加回车符。它们在磁盘上的存放格式是:

字符	7		A	T		O	N	E		B	L	O	W	,	O	P	
ASCII 码	55	13	65	84	13	79	78	69	13	66	76	79	84	44	79	80	13
字符号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

字段号: 0 (字符号 0–1), 1 (字符号 2–4), 2 (字符号 5–8), 3 (字符号 9–13), 4 (字符号 14–16)

记录号: 0 (字符号 0–1), 1 (字符号 2–4), 2 (字符号 5–8), 3 (字符号 9–16)

这是字节号、字段号和记录号都是从 0 开始编号。顺序文件中记录的长度和文件长度可以是任意的。

6.2.2 打开顺序文件(OPEN)

在存访文件之前,应打开磁盘文件。磁盘文件打开之后,DOS 就会检索有关该文件的信息。例如,文件是否在磁盘上,如果在的话,在盘的什么位置。打开文件的命令格式如下:

OPEN FILENAME 〔,Ss〕〔,Dd〕〔,Vv〕

命令参数说明:

FILENAME: 文件名以字母打头的字符串组成(不能含有逗号),最多可有 30 个 ASCII 字符。

Ss: 设定软盘驱动器所在的槽口号,CEC- I 型机定为 6。

Dd: 设定软盘驱动器在软盘驱动器接口卡上的设备号,d 值必须为 1 和 2,CEC- I 型机 d=1。

Vv: 软盘片的卷号,一张软盘片只能有一个卷号,卷号值 V 可以为 1- 254 之间的整数卷号是在用 INIT 命令进行盘片初始化操作时就已确定。不可更改,起着核对盘片的作用。

若在给定的软盘上没有该文件,那么,DOS 将在磁盘目录中添补一项新的文件。当执行 OPEN 命令之后,DOS 将给打开的文件分配 595 个字节的文件缓冲区,供磁盘与主存之间信息交换用。

下面这个程序,只是在磁盘生成顺序文件。

SEQUENTIAL:

```
100  D $ =CHR $ (4)
200  PRINT D $;"OPEN SEQUENTIAL"
300  END
```

我们称这个程序为"program1"(程序 1)。这个程序执行后,目录中就有了 SEQUENTIAL 文件。可以打入 CATALOG 命令,以便证实,这时,显示的目录如下:

```
DISK VOLUME 254
*A  002  HELLO
```

T　001　SEQUENTIAL

由上看到，在 SEQUENTIAL 的扇区数之前，有个英文字母 T。它表示 SEQUENTIAL 是一个文本文件。

6.2.3　关闭顺序文件(CLOSE)

实际上，程序 1 并不是一个很好的程序，因为此程序结束之后文件没有关闭。这里要说的 CLOSE 关闭命令是非常重要的。在对某一文本文件完成读／写操作后必须及时关闭该文件，使得 DOS 命令收回分配给该文件的文件缓冲区，否则未关闭的文件会引起数据的丢失。

CLOSE 命令有两种格式，第一种格式:

CLOSE

命令中不带任何参数，它把整个软盘中所有打开着的文件统统关闭。

有时，可能要关闭一个或几个指定的文件，那么就应该给 CLOSE 命令加上文件名，其格式为:

CLOSE　FILENAME

在 CLOSE 命令中，可以不用槽号，驱动器号及磁盘号，因为文件已经打开了，DOS 会知道打开文件在什么地方。

程序 1 应该加上:

```
290  PRINT D $;"CLOSE"
```

或者加上:

```
290  PRINT D $;"CLOSE SEQUENTIAL"
```

6.2.4　把信息写入顺序文件中(WRITE)

将信息送到磁盘和送到屏幕的方法类拟，都是使用 PRINT(打印)语句。任何可以打印的内容，都可以存入磁盘文件。实际上，可以把顺序文件设想为电视屏幕，甚至把它当作打印机的一张纸更好一些。

当文件上打入一些东西时，DOS 就要修正内部指针，指向数据将要存入的下一个盘面位置，就好象打印纸向前走纸到下一行一样。顺序文件的指针只能向前移动，而 OPEN 命令可把指针移回到文件的开始处。

把数据打入到磁盘文件之前，必须先使用 WRITE(写)命令通知 DOS，后面的 PRINT 语句是要写入文件，而不是在屏幕上显示。对于

顺序文件来说,WRITE 命令的格式是:

WRITE FILENAME〔,Bb〕

WRITE 命令发出以后,紧接着,输出就直接送到指定的文件中。必须注意,这种输出中还可以包含着程序执行中的错误信息,不过,当文件上存有错误信息后,WRITE 命令就被消除了。这时在屏幕上只能看到 BASIC 提示符和光标。参数 Bb 指明从顺序文件的第 b 个字节开始写数据,如果参数 Bb 省略,则从文件的第 0 个字节开始写数据。

例: 在程序 1 中加入下面二行:

```
210  PRINT D $;"WRITE SEQUENTIAL"
220  PEINT " THIS TEXT WILL BE STORED IN THE
      FILE"
```

程序就可以做下列事情:

① 生成一个文件

② 把该文件打开

③ 把文本资料存到文件中去

④ 关闭该文件

请记住,每次执行这一程序时,凡是在 PRINT 语句中的程序皆会修改和删除原来文件中的数据。若 PRINT 语句的字符数比原来文件的字符数少,则修改之后,在新文件的尾部还跟着没有修改的老文件。为了消除残留的数据,在存入新数据之前,要先把原文件消去。把 DLELTE 命令放在 OPEN 命令的前面,可以做到在每次执行程序时,把原文件删去,然后由 OPEN 命令重新送入。

但是,若在磁盘中没有一个名叫 SEQUENTIAL 的文件,而要执行此程序,这时,将会看到出错显示 FILE NOT FOUND。为了防止类似的错误发生,可以在 DELETE 命令前,再加一个 OPEN 命令。

程序 1 作了上述修改后,则可列出如下程序:

```
100   D $ =CHR $ (4)
110  PRINT D $;" OPEN SEQUENTIAL"
120  PRINT D $;" DELETE SEQUENTIAL"
200  PRINT D $;" OPEN SEQUENTIAL"
210  PRINT D $;" WRITE SEQUENTIAL"
220  PRINT " THIS TEXT WILL BE STORED IN THE
```

```
          FILE"
290  PRINT D $;"CLOSE"
300  END
```

6.2.5 读取顺序文件(READ)

如同输出可以直接存到磁盘一样，输入也要以直接取自磁盘文件。DOS 的 READ 命令指明磁盘文件是输入数据的来源。其命令格式如下：

READ FILENAME〔,Bb〕

该命令将对已打开的文件 FILENAME 执行读操作，参数 Bb 指明了从顺序文件的第 b 个字节开始读数据，如果参数 Bb 省略则从文件的第 0 个字节开始读数据。执行 READ 命令以后，随后所有的 INPUT 语句或 GET 语句都将从文件中读取数据，而不是从键盘上读数据，一直持续到下一个 DOS 命令，或者是由一个出错把读命令消掉为止。

注意：在用 INPUT 语句读入数据时，一次读一个记录，并把这个记录中以逗号分隔的数据串依次赋给 INPUT 语句中的各个变量。如果变量的类型和所给的数据类型不匹配，则给出?REENTER 信息，继续取下一个数据；如果记录中的字段数多于 INPUT 语句中的变量个数，则给出信息：EXTRA IGNORED，忽略多余的数据部分；如果记录中的字段个数少于 INPUT 语句中变量的个数时，屏幕将出现??信息，自动到下一个数据域中取数据。

下面的程序 2 说明了 READ 命令的使用。

```
100  D$ =CHR$(4)
110  INPUT "FILENAME TO READ:";F$
120  PRINT D$;"OPEN";F$
130  PRINT D$;"READ";F$
140  INPUT A$
150  PRINT A$
160  GOTO 140
170  END
```

程序 2 把磁盘中由程序 1 生成的文本文件数据读出后显示出来。

当文件内容全部输出之后，在屏幕上显示出 END OF DATA(数据完毕)，并在程序执行停止时，显示出 BREAK IN 140。

由上面程序的执行情况可以看出，程序 2 不是一个好的编程实例，它会打印错误的信息，并且还会在执行中停机。为了防止 END OF DATA 的错误，可在出错之前进行状态检测。当检测到:“数据完毕”时，程序 2 可以转移控制 CLOSE 命令。检测“数据完毕”的最简便方法是用 ONERR GOTO 语句。另一种方法是改写程序 1，即在整个程序打完继而打入 END 语句时，再打入一个特别的字或字符作标志。然后，修改程序 2，以便找到文件的结束标志，并恰当地关闭文件。

6.2.6 使文件添加内容(APPEND)

要想在文件的结尾处加入数据，必须先读出文件的结尾。当然也可以读出文件的每一项，直到找到最后的结尾为止。但如果这个文件是大型文件，就非常浪费时间，用 APPEND 命令就可以完成这一工作。其命令格式如下:

APPEND FILENAME

该命令将自动打开顺序文件 FILENAME，并把文件位置指针移到文件的末尾(即文件最后一个字符的后面)。在 APPEND 命令之后必须使用 WRITE 命令，如果使用了 READ 命令，执行 INPUT 语句时，则会显示信息 END OF DATA。如果在执行 APPEND 命令后写入数据，那么，已有文件后边会加上新的数据。

APPEND 命令虽可以代替 OPEN 命令，但它们之间有着明显的不同:

① 使用 APPEND 命令之前，一定要先有文件，若没有文件，就会显示“FILE NOT FOUND”(文件找不到)。因为 APPEND 命令不会生成文件，只是有文件存在时，它才可以使用文件。

② APPEND 命令是把指针放在文件的结尾，而 OPEN 命令则把指针放在文件的开头。

6.2.7 文件的定位(POSITION)

顺序文件通常从文件的前面顺序读写。但是，若使用

POSITION 命令,就可以读写文件的任意记录。命令格式如下:

POSITION FILENAME 〔,Rr〕

该命令用来对顺序文件执行定位操作,它可以把文件位置指针从当前位置移到下面第 r 个记录的起始位置。参数 Rr 指明的是当前位置的第 r 个域。当参数 Rr 省略时,表示不移动文件的位置指针。

在使用 POSITION 命令之前,必须先把文件打开。文件名 FILENAME 须与 OPEN 命令中的文件名一致,它和其它 DOS 命令一样,将撤消 WRITE 和 READ 命令的功能。所以,POSITION 命令必须用在 WRITE 和 READ 命令之前,否则,使用过 POSITION 命令之后还需要重新使用 WRITE 或 READ 命令,才能执行写或读操作。

6.3 随机文本文件的管理

6.3.1 随机文本文件的存放格式

顺序文本文件在磁盘上的存放格式是一个接着一个,每个记录以回车符为界,记录长度是不固定的。因此要想随机地使用或修改每个记录时,要一个个数回字符的个数或字节的个数,这是很不方便的。

随机文件则不同,它在磁盘上的存放格式总是以记录为单位的,而且每个记录都有固定的长度。一个记录的最大长度是 32767 个字符。

现在假定有下列一些数据存在一个随机文件上,记录长度为 5:

7 AT ONE BL,W

那么它们在磁盘上的存放格式是:

字符	7					A	T				O	N	E		
ASCII 码	55	13	00	00	00	65	84	13	00	00	79	78	69	13	00
字节号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

字段号　0　　0　　0

记录号　0　　1　　2

字符	B	L	,	W						
ASCII 码	66	76	44	87	13	00	00	00	00	00
字节号	15	16	17	18	19	20	21	22	23	24
字段号		0		1						
记录号			3							

随机文件用一个序号表示记录在文件中的绝对位置。每个文件的第一个记录号用序号 0,下一个记录是序号 1,接着是序号 2,依此类推。在程序中必须指明,随机文件中被选中的是哪一个记录以及对记录中的哪一部分进行操作。

使用随机文件,可以实现快速地查找,增补或修改其中任一记录中的数据,而不影响其它记录中的内容。

6.3.2 打开和关闭随机文件

打开随机文件的命令格式如下:

OPEN FILENAME,Lj

若要定义一个随机文件,当文件打开时,还要包括一个附加参数 Lj,j 是 1 ~ 32767 之间的数,Lj 长度参数用来指明每个记录的长度。

在打开随机文件时,DOS 并不把该文件的记录长度存入磁盘,如果我们重新打开 FILENAME 文件时,参数 Lj 所指明的长度与创建时的长度不一致,则 DOS 将按照给的参数来计算文件中各记录的位置,这样读写随机文件时可能会造成混乱的结果。为了避免出现这种情况,我们必须保存好关于随机文件的结构和数据格式,以便在编制检索程序时不至于搞错。最为简单的办法是把这些信息作为文件的第 0 个记录保存起来,而把记录长度附在文件名中。如:MAILER 文件可取名为 MAILER.L30。

随机文件的关闭命令 CLOSE 与顺序文件的关闭命令相同,可参考 6.2.3 一节的说明。

6.3.3 随机文件的读写

读写随机文件的命令格式如下:

READ FILENAME〔,Rr〕〔,Bb〕

WRITE FILENAME〔,Rr〕〔,Bb〕

这里有二个参数 Rr 和 Bb，分别用来指明是对随机文件中第 r 个记录执行读／写操作，其起始位置为第 b 个字节。

参数 Rr 指定文件的第 r 个记录，r 值可以是 0 ～ 32767 之间的整数，当 r=0 或省略时，表对 FILENAME 文件的第 0 个记录执行读／写操作。

参数 Bb 表示从该记录中第 b 个字节开始执行读／写操作。如果 b 值很大，超出记录长度，则在使用 PRINT 语句后，写入的数据会破坏下一个记录中的内容。

① 在使用随机文件读写命令应该注意的是：在随机文本文件中，任何一个向文件输出的 PRINT 语句，必须跟在一个 WRITE 命令之后；同样，任何一个从文件读取数据的 INPUT 或 GET 语句，必须跟在一个 READ 命令之后。并且通过 WRITE 或 READ 中的 Rr 和 Bb 选项指定读写记录号和字节号。否则文件实际被当作顺序文本文件处理。

② 在使用读／写命令后，不能使用其它的 DOS 命令。另外 INPUT 语句或 GET 语句会终止 WRITE 命令的执行，PEINT 语句终止 READ 命令的执行。

6.3.4 随机文本文件的实例

假定某单位有 1000 人，想把每个人的姓名，性别、籍贯，基本工资等数据存入一个文本文件中。首先我们可以将这些人的各项数据写在 DATA 语句里，通过循环语句，用 READ 将这些数据读入有关数组(当然这些数据也可以用 INPUT 语句从键盘上输入)。下面的这个程序将建立一个随机文本文件 BCD:

```
10 DATA "MAO ZHEN- HUA","NAN","BEIJING",
   107.5
20 DATA "LI WEI- GUO","NAN","SHANGHAI",79
30 DATA "WANG LIANG","NAN","TIANJING",69
40 DATA "LU HUA","NU","BEIJING",74.5
50 DATA "WEN LING","NU","HEBEI TANGSHAN",56
```

```
·
·
·
1000  DIM A $ (1000,3),B(1000)
1010  FOR I=1 TO 1000
1020  FOR J=1 TO 3
1030  READ A $ (I,J)
1040  NEXT J
1050  READ B(I)
1060  NEXT I
2000  D $ =CHR $ (4)
2010  PRINT D $;"OPEN BCD,L100"
2020  FOR I=1 TO 100
2030  PRINT D $;"WRITE BCD,R"; I- 1
2040  PRINT A $ (I,1);",";A $ (I,2);",";A $ (I.3)
      ;",";B(I)
2050  NEXT I
2060  PRINT D $;"CLOSE"
2070  END
```

运行这个程序后,将在磁盘上建立一个名为 BCD 的随机文本文件,共 1000 个记录,每个记录都是 100 个字节,每个记录有 4 个字段。

请注意:若将上述程序中修改如下两行:

```
2020  PRINT D $;"WRITE BCD"
2030  FOR I=1 TO 1000
```

运行后得到的 BCD 文件,将是一个顺序文本文件。

如果要改写 BCD 文件中的某个记录(如 501 号),可运行下面的程序:

```
10  D $ =CHR $ (4)
20  PRINT D $;"OPEN BCD,L100"
30  PRINT D $;"WRITE BCD,R501"
40  PRINT      (被改写的内容)
50  PRINT D $;"CLOSE BCD"
```

被改写的内容可写在40行的PRINT语句中。改写时最好改写整个记录(如果记录又分段,要强行加上逗号)。这样改写后的记录不会留下原来记录的多余部分。如果要改写某个记录的某一字段,可找出此字段的首字符在该记录中的相对位置,用Bb加以设定。

针对上述方法建立的BCD文件,写一个检索程序是很方便的。比如我们查找工资在100元以上的人员名单,可运行下列程序:

```
10  DIM A $ (1000),B(1000)
20  D $ =CHR $ (4):N=0
30  PRINT D $;"OPEN BCD,L100"
40  FOR I=1 TO 1000
50  PRINT D $;"READ BCD,R";I- 1
60  INPUT A $,B $,C $,D
70  IF D>=1000 THEN N=N+ 1:A $ (N)=A $:B(N)=D
80  NEXT I
90  PRINT D $;"CLOSE BCD"
100  FOR I=1 TO N
110  PRINT A $ (I),B(I)
120  NEXT I
130  END
```

6.4 EXEC执行文件

6.4.1 EXEC执行文件

DOS系统中的EXEC命令可以使CEC- I的控制权由键盘控制变为文件控制。也就是说,我们可以把一些命令或者程序输出到一个顺序文件中去。当这种文件建立好以后,用EXEC命令来执行它,这时从文件中读回内存的信息,如同从键盘上输入的一样,如果是立即执行命令,就执行它;若是带行号的程序,就把它们送内存的程序区。我们把这种命令或程序文本文件称为EXEC执行文件。

让我们先看一个例子,假定有如下一个程序:

```
10  PRINT "*  TIMES  *"
20  PRINT "APPLE:";TAB (10);0.35
30  PRINT "PEAR:";TAB (10);0.28
```

```
40  PRINT "ORANGE:";TAB (10);0.46
50  END
```

运行这个程序会显示出三种水果的名称和价格。假定以 FRUIT 为程序名，可以先把它存在磁盘上。

现在，如果我们打算依次做以下三件事：

① 运行 FRUIT 程序。

② 把内存中当前程序的 20 行至 40 行列出来。

③ 把当前磁盘言件目录列出来。

通常我们是依次敲入下面三条命令分别实现的：

```
RUN FRUIT
LIST 20,40
CATALOG
```

如果把这三条命令作为程序输出的数据，存入到顺序文件中，例如取文件名 IT，这件事也可以通过下列程序的运行来实现：

```
10  D $ =CHR $ (4)
20  PRINT D $;"OPEN IT"
30  PRINT D $;"WRITE IT"
40  PRINT "RUN FRUIT"
50  PRINT "LIST 20,40
60  PRINT "CATALOG"
70  PRINT D $;"CLOSE"
80  END
```

这时再使用 EXEC 命令：

```
EXEC IT
```

这个 DOS 命令会自动打开名为 IT 的文件，并从中顺序将一个个记录送入内存(好象从键盘上输入的一样).然后再一条一条地执行这些送入内存的命令。

6.4.2 EXEC 命令的使用

EXEC 命令的一般格式为：

```
EXEC FILENAME 〔,Rr〕
```

该命令将从顺序文件 FILENAME 中的第 r 个记录开始执行该文

件中所含的命令。这些命令就象是从键盘上一句一句打入的一样。在EXEC命令中,参数Rr省略则表示从文件的开头去执行。EXEC执行文件中所含的命令可以是BASIC命令,监控命令和DOS命令。

在使用EXEC命令时要注意以下几点:

① 由EXEC打开的文件,只有在它的全部信息输入内存之后(如果这些信息是命令,则要全部执行完以后)才自行关闭。CLOSE命令是关闭不住它的。并且在任何时刻只有一个EXEC打开一个文件。如果EXEC中以有一个EXEC时,关闭掉第一个EXEC打开的文件,由第二个EXEC打开另一个文件。

② EXEC中若有RUN命令,EXEX命令会耐心地等待RUN执行完毕后,再继续执行EXEC文件中的下一个命令.但要注意,在EXEC命令下运行程序时,程序中的任何INPUT语句都会将EXEC文件的一个记录作为输入数据,而不从键盘上输入数据。

③ 如果在EXEC执行文件中含有带行号的BASIC程序行,EXEC命令并不执行这些程序得中的命令.而是按其行号次序将它们装入内存中,就象这些BASIC语句行是从键盘上输入到内存中一样。

④ EXEC命令不会被CTRL- C中止。

6.4.3 EXEC执行文件的实例

假定内存中已有一个BASIC程序,我们对其中的某一段程序(比如2000行列10000行)很感兴趣,那么可以通过运行下面的程序,把这段程序保存到一个顺序文本文件上(比如文件名为LISTING)。

```
1  D $ =CHR $ (4)
2  PRINT D $;"OPEN LISTING"
3  PRINT D $;"WRINE LISTING"
4  POKE 33,30
5  LIST 2000,10000
6  PRINT D $;"CLOSE"
7  END
```

在这个程序里,由于第3行的作用,使这些程序不是送往屏幕,而是送给文件LISTING了.另外,用LIST将程序送屏幕显示时,并未占满屏幕,而是从左边开始向右.行宽只有30个字符位置。因此用了第

4 行的改变窗口宽度命令，以避免存入文件时在每个终端行上加入 10 个多余的空格符。

当如上的 LISTING 文件建立好以后，执行下列命令：

EXEC LISTING

会将 LISTING 中的 2000 到 10000 行程序装入内存中，为其它程序使用，而不必从键盘上输入。

我们可以利用 EXEC 的这一特性，把某个子程序插入到主程序中，修改部分程序，或把几个程序合并为一个程序.可能读者会问，把那些有用的程序段用 SAVE 命令存入磁盘，以后 LOAD 出来，不也可以为别的程序用吗?这当然是可以的.但用 EXEC 运行一个存有程序行的文件，要比 LOAD 功能强得多，至少有以下几个优点：

① 当一段新的程序输入内存后，再 LOAD 出一些有用的程序行，将会把已输入机器的程序冲掉.而 EXEC 命令从文件读入的程序行则会加在内存中原有的程序中。因为 EXEC 命令读入的内容，相当于从键盘上敲入的一样。

② LOAD 进入内存的程序段，只能来源于一个 BASIC 程序.因为你没有办法将两个源程序存入一个程序名中.而 EXEC 运行的文本文件，其数据(在此为程序行)可以来自不同的 BASIC 程序.因为一个文件建好以后，还可以再次增加数据.这样一个文本文件可以收集多个程序的有用部分，经 EXEC 后，在内存中构成一个新程序。

③ 浮点 BASIC 和整数 BASIC 分属两种不同的文件类型，通过 SAVE 和 LOAD 命令，不会改变任何一个 BASIC 程序的类型属性。而通过建立文件的文法，将某些程序行写入文件后，它们的属性为文本文件类型，失掉了 BASIC 的属性.用 EXEC 把它们装入内存时，当时系统属性就是它们的属性。

6.5 机器语言磁盘文件

6.5.1 机器语言文件的格式

DOS 操作系统除了可以对 BASIC 程序文件，文本文件进行操作处理外，还能够对机器语言及二进制映象文件进行操作.在文件目录中机器语言文件的类型标志用字母 B 表示。机器语言文件存放的是主机内存中的二进制信息副本，即把主机内存中某个地址段的信息以文件的

形式保存在磁盘上。这些文件信息可以是机器语言程序，也可以是二进制数据或图象信息等。例如低分辩率和高分辩率的两种图象，都可以存入磁盘，以备重新调用和显示。通常机器语言程序文件可以直接调入内存，并予执行，或者由 BASIC 程序用 CALL 语句或 USR 函数调用。

机器语言文件在磁盘上的存储格式为：

起始地址　　文件长度　　二进制数据块

DOS 有三个专为机器语言文件设计的命令，它们是 BSAVE，BLOAD，BRUN。这三个命令的作用与非二进制的 SAVE，LOAD，RUN 等效。

6.5.2 把二进制内容存入磁盘（BSAVE）

BSAVE 命令就是把二进制的映象存到磁盘中去。其格式如下：

BSAVE FILENAME，Aa，Lj

该命令将把内存中指定的一段二进制数据，以 FILENAME 为文件名，保存到磁盘上。

参数 Aa 是地址参数，它指出要存储的二进制数据在内存中的起始地址。该地址可以是 10 进制或 16 进制的常数。如果是 16 进制数，a 的前面要用符号“$”；若是 10 进制数，a 的值必须在 0～65535 之间，不允许有负的数值。

Lj 参数指明了该二进制数据块的字节长度。j 也可以用 10 进制或 16 进制表示。j 的值可以在 0～32767 之间。若想存入大于 32767 字节的内容，那么要使用两个 BSAVE 语句。

例如：用下面的二个命令都可以在磁盘上创建一个名为“PICTURE”的二进制文件，该文件将包含在主机内存中高分辩图象第二页内的图象信息。

BSAVE PICTURE，A$4000，L$2000

BSAVE PICTURE，A16384，L8192

6.5.3 二进制文件装入内存（BLOAD）

BLOAD 命令是从磁盘上取出二进制文件的内容，并把它装入内存。其命令格式如下：

BLOAD FILENAME〔，Aa〕

在该命令中，地址参数 Aa 可以任选。如果选用地址参数则指用了二进制文件将装入到内存区域的起始地址。a 的值可以在 0 ~ 65535 之间(16 进制则为 0 ~ FFFF)。如果不选地址参数，那么二进制内容装入内存的起始地址就是它以前存入磁盘时所指定的地址。若是把机器语言程序装错了地址，程序可能不会正常工作。

BLOAD 命令与 LOAD 命令不一样，BLOAD 命令执行时，系统工作状态不会发生变化，也不会破坏内存中的 BASIC 程序，除非被装入的二进制文件侵入到 BASIC 程序所占用的存储空间。

例如：若要将 PICTURE 文件装回到原来的存储区中(即高分辩图象区的第二页)中，可用命令：

BLOAD PICTURE

BLOAD PICTURE,A $4000

BLOAD PICTURE,A16384

使用下列命令则将图象信息装入高分辩图象区的第一页：

BLOAD PICTURE,A $2000

BLOAD PICTURE,A8192

6.5.4 运行二进制程序(BRUN)

若磁盘上有一个机器语言程序文件，可以直接用 BRUN 命令装入内存并执行该程序。其命令格式如下：

BRUN FILENAME〔,Aa〕

BRUN 命令与 BLOAD 命令几乎等同，只是 BRUN 命令在文件已经装入内存之后，要跳转(JMP)到该文件的第一个字节去执行程序。

在使用 BRUN 命令时，应该注意以下几点：

① 必须确保 FILENAME 是由机器语言程序所组成的二进制文件。对于二进制图象文件，绝不可使用 BRUN 命令，因为它的结果难以预测。

② 运行程序的入口地址是该文件中的第一个字节。

③ 如果在命令中使用了参数 Aa，则 a 所给出的地址要和建立文件时起始地址相一致。

6.6 DOS 的辅助命令

6.6.1 使用 DOS 辅助查错

磁盘文件程序一般都比较复杂，DOS 提供了两个命令可以帮助“诊断”程序，即 MON 和 NOMON。

(1) **置监视方式(MON)**

在用程序方式调用 DOS 命令时，DOS 命令及主机与磁盘之间的信息交换均不在屏幕上显示出来，但在调试程序时，监视这些信息却是十分必须的，通过它我们可以找出程序中的问题。MON 命令可以满足我们监视这些信息的要求。其命令格式如下：

MON 〔C〕〔,I〕〔,O〕

在该命令中有三个参数可供我们选择，其中 C 表示显示所使用的 DOS 命令，I 表示显示由磁盘读入主机的文本信息，O 表示显示由主机输出到文本文件中的信息。

这三个参数可以按任何顺序和任意组合的形式出现，如果这三个参数均不出现，则 MON 命令不起任何作用。

MON 命令执行之后，始终保持有效，直到执行了 NOMON，INT，FP 命令，或者重新启动 DOS 为止。

(2) **解除监视方式(NOMON)**

NOMON 命令的作用是取消 MON 命令。NOMON 命令也要使用三个类似于 MON 命令的参数：C，I，O。解除监视是指出哪一个数据不要受监控。例如，假定已经发出过 MON I，O，C，命令，现在用

NOMON O

命令，就会把输出到磁盘的监视取消，而从磁盘输入到内存和 DOS 命令监视仍然有效。

6.6.2 设置文件缓冲区(MAXFILES)

MAXFILES 命令用来指定一次打开文件的最大数目，其格式如下：

MAXFILES n

其中参数值 n 可以是 1 ～ 16 之间的整数。

每个文件在内存中占用了 595 个字节存储单元作为缓冲区。每个缓冲区中又有二个 256 字节的段，一个用于读，另一个用于写；其余的

83 个字节用来存放该文件在磁盘上的管理信息。对某个文件执行读操作时,DOS 首先从盘上将一个扇区的信息(256 字节)读入到该文件的缓冲区中,然后再将其中的一部分送给程序。如果对某个文件执行写操作时,也是把信息送到文件缓冲区,待写满 256 个字节时再送到磁盘上。

文件缓冲区的个数决定了程序一次可以打开的文件个数。启动 DOS 时,系统自动执行了“MAXFILES 3”命令,把文件缓冲区的数目定为 3 个。所以它一次最多可以打开的文件个数只能是 3 个。如果一次打开的文件个数大于 MASFILES 命令中指定的参数值 n 时,DOS 将给出出错信息:NO BUFFER AVAILABLE。

在执行 MAXFILES 命令时,系统将修改 HIMEM:的值,所以放在 HIMEM 之下的字符串变量有可能被破坏。故在使用 MAXFILES 命令时,应把它作为程序的第一条语句,放在任何串变量赋值语句和字符串数组定义之前。

6.6.3 CEC- I BASIC 程序的链接

在 DOS3.3 操作系统中没有提供 CEC- I BASIC 程序之间的链接操作,但在 DOS3.3 系统主盘上提供了 CHAIN 程序,利用该程序,可以完成程序之间的链接。

使用时首先将 CHAIN 程序复制到我们的工作盘上,然后在第一个程序的末尾加上以下两条语句即可:

```
PRINT CHR $ (4);"BLOAD CHAIN,A520"
CALL 520 “第二个程序名”
```

第七章
打印机输出控制

如果在你的CEC－Ⅰ中华学习机系统中能够配上一台打印机，系统就更加完善了。这样，在你设计程序使用CEC－Ⅰ时就会感到非常方便。通常我们把打印机的输出(打印的内容)称为“硬拷贝”，通过打印机，你可以把程序清单，文件报表、出错信息，甚至图形打印出来，作为永久的记录。

打印机通过打印机接口与主机系统相连接后，成为受控于主机系统的一个外部设备。它具有接收并处理由主机送来的控制命令和打印数据的能力。通过打印机对这些命令的执行和处理，可以控制和改变打印数据的格式，走纸量和字符形状等等。还可以将输出的数据作为图象信息进行打印，以达到打印汉字的目的。

CEC－Ⅰ中华学习机可以配备多种型号的打印机，常见的有CP－80，RX－80，FX－80，YAMATO，STAR等等。打印机的种类虽然很多，但其操作方式却大同小异。本章中，我们以MS－80型打印机为例，详细地说明如何操作和控制打印机。其它型号的打印机请参考随机所带的使用说明。

对打印机的操作方式通常分为手动方式和程序方式，下面我们分别加以说明。

7.1 打印机上的控制键

一般在打印机的侧面有一个标有ON／OFF的电源开关(有些打印机用打点位置表示电源开，在打印机的上方或前面板上有一排按键和指示灯，用来控制打印机的操作和指示其工作状态。下面是一些常用的按键和指示灯的功能：

按键	功能
ON／OFF	电源开关

SELECT(带指示灯)	暂停或重新开始打印机操作
FF(From Feed)	换页
LF(Line Feed)	换行

指示灯	功能
POWER	打印机电源接通
READY	打印机准备好,可以打印资料
PAPER OUT	没有纸了

打印机通常有自己的电源,因此,在使用打印机之前要注意是否已经把电源接好,然后再打开电源开关。一般的打印机都有一个标着 POWER 的电源指示灯,亮了就表示打印机已经通电了。另外一个重要的按键就是标有 SELECT 或 ONLINE 的按键,当这个按键上的灯亮时,就表示打印机由 CEC- Ⅰ 控制,而当你按下这个按键时,灯就熄灭,打印机的动作也会停止,这时就可以用手动方式来控制打印机,再按一次这个按键,指示灯又会亮起来,进入由 CEC- Ⅰ 控制的状态。

另外一个常用的指示灯就是 READY,这个指示灯用来指示打印机的工作状态。一旦打印机工作完成,可以接受下批印出的资料时,这个灯就亮;当打印机正在工作时,这个灯是熄灭的。

有些打印机上有一个 PAPER OUT 的指示灯,当打印纸用完或没有装好时,这个灯就会亮起来。另外,打印机里面还有一个小扬声器,当打印机出现故障时(如色带断了,或卡住了,或没有纸了),就会发出"嘟"的声音,指示你来排除故障。

大部分的打印机都有一个 FF 按键,用来进行打印纸的换页操作。在进行换页时,首先要进入手动操作状态(按 SELECT 键,使灯熄灭),然后再按 FF 键,就会把打印纸卷到下一页的第一行上。另外一个重要的按键,就是 LF 键,用来对打印纸进行换行的操作。使用这个键,同样需要进入手动操作状态,按 LF 键会使打印纸空走一行。

关于打印机使用详细的操作,可以具体参考自己使用的打印机使用说明书

7.2 打印机的连通与使用

当 CEC- Ⅰ 在工作时,一般所产生的输出信息都会在屏幕上显示出来,而

当你使用打印机时，CEC－Ⅰ只不过是把打印机看成是屏幕的"拷贝"。也就是说，当你想使用打印机打印输出结果时，只要告诉CEC－Ⅰ把屏幕的输出送到打印机上就行了。CEC－Ⅰ连通打印机的命令为：

]PR#1

在这条命令中PR#的功能是把CEC－Ⅰ的输出送到某个指定的扩充槽口上，一般打印机接口套是设定在1号槽口上，所以执行的命令应该是CEC－Ⅰ。执行完这条命令以后，CEC－Ⅰ在屏幕上的所有输出内容，都会通过打印机打印出来，其打印的格式也会与屏幕上显示的一模一样。

使用这条命令要注意的是，如果打印机电源开关在执行命令之前没有打开，或者是打印机没有准备好可以接受打印信息，或者是扩充槽上根本没有打印机接口套，那么CEC－Ⅰ就会进入等待状态。

如果你的打印机已经连通好了，那么现在就可以下达CATALOG命令，这样你就会同时在屏幕和打印纸上得到一份磁盘文件的目录，如果你下达LIST命令，就会在打印机上得到一份程序清单。

CEC－Ⅰ使用的打印机通常是能够在一行上打印80个字符，但我们在列程序清单时，却发现在打印出的一行上最多只有40个字符。这是因为CEC－Ⅰ控制方式的缘故，通常只会印出与屏幕上格式完全相同的40个字符而已。所以要让打印机在一行上打印出超过40个字符时，就要下达特殊的命令。其命令为：

]PR#1

]POKE 1657,80

]LIST

在POKE命令中的80表示在一行中要印出80个字符。在80个字符打印状态下，输出的字符在屏幕上是不显示的(大于40个字符也一样)，当打印工作完成以后，光标在屏幕上闪烁时，就表示所有的资料都印完了。要回到40个字符以及屏幕显示状态，只要用下面的命令：

POKE 1657,40

就可以了。

当打印完所需要的资料以后，希望打印机不再有作用，就可以用下面的命令把打印机与CEC－Ⅰ的连通断开：

]PR#0

另外，当同时按下 CTRL- RESET 这两个键以后，也会自动解除打印机与 CEC- I 的连通。

7.3 在程序中使用打印机

7.3.1 在程序中连通打印机

PR#1 命令不但可以在 BASIC 提示符下直接使用连通打印机，而且也可以放在程序中做为一条语句使用。在程序中一旦执行了 PR#1 语句，那么就会连通在1 号槽上的打印机，输出的资料就会送到打印机上印出来。反过来，PR#0 语句会解除打印机的输出。

下面是一个在没有 DOS 系统下使用打印机的例子：

```
10  REM OUTPUT 2 LINES TO A PRINT
20  PR#1
30  PRINT "CHINA EDUCATION COMPUTER"
40  PRINT "SHANNXI COMPUTER FACTORY"
50  PR#0
60  END
```

上面的例子中，20 行连通打印机，30 和 40 行打印出两行英文字母，然后在 50 行解除打印机的作用。如果在有 DOS 系统的情况下，那么在语句的写法上有点区别，这时 PR#1 就变成了 DOS 命令，在程序中要使用 DOS 命令的话，第一个字符必须是 CTRL- D，即 CHR $ (4)，于是程序中的 20 行和 50 行语句就得改成

```
20  PRINT CHR $ (4);"PR#1"
50  PRINT CHR $ (4);"PR#0"
```

7.3.2 打印字符的定位控制

程序中打印机的使用与在屏幕上的使用基本相同，但也有一些区别。比如说打印机的走纸只能向一个方向移动，所以就没有办法用 VTAB 来上下地定位移动；打印机在打印一行时，回车符没有发出之前，我们可以使打印头向左或向右移动，但是一旦送出回车符或换行命令后，打印头就无法回到上一行去了。

使用打印机时，PRINT 语句中的逗号(，)和 TAB 函数常常会得

出一些很奇怪的结果,所以我们尽量不去使用它。通常是利用 SPC 函数和 HTAB 语句对一行中打印的字符进行定位。下面是一个使用 HTAB 语句定位的程序。

```
LIST

100 D$ =  CHR$ (4)
110  PRINT D$;"PR#1"
120  PRINT
130  FOR I = 1 TO 39 STEP 5
140  HTAB I
150  PRINT "<><>";
160  NEXT I
170  PRINT
180  PRINT D$;"PR#0"
190  END

]RUN
```

7.4 打印机的控制命令

每种型号的打印机都有它特有的功能,这些功能显然要比 SPC 函数或 HTAB 语句等更有用些。通常 CEE- I 并不会了解这些控制功能是什么,而需要使用者事先了解所使用打印机的特性,知道如何控制它。这样就可以在直接方式下或在程序中送出控制打印机的命令。这些控制命令通常是一些控制符号或 ESC 字符串,在 BASIC 程序中常用 PRINT 语句送给打印机,以达到控制,指挥打印机的目的。

7.4.1 基本动作的控制

首先我们看几个最基本的控制动作,这些动作如表7.1 所示。

在这里要说明的是,CR 的原意是把打印头移回到最左边,纸张并没有走一行,所以如果再打印一行就会和上一次打印的内容重叠了。而 LF 只是让纸张走一行,打印头不一定移回到左边。但是一般打印机出厂时被设置成自动换行,也就是说,用 CR 就等于是 CR 加上 LF。

我们可以用下列命令:

POKE　1529,255

把自动换行功能暂停,这样当你送出 CR 命令时只是把打印头向左移,而需要换行时,必须送出 LF 命令。要恢复自动换行功能,需要用下面的命令:

POKE　1529,0

表7·1 打印机基本控制动作

SACII 符号	控制字符	十进制代码	纸张动作	控制功能
BELL	CTRL-G	7	无	响铃,使打印机鸣 3 秒
BS	CTRL-H	8	无	退格,从打印缓冲区删去一个字符
LF	CTRL-J	10	进一行	完成打印,并走纸一行
FF	CTRL-L	12	跳页	走纸到下一页的页首
CR	CTRL-M	13	进一行	送回车符,完成打印缓冲区数据打印

下面的程序为你示范 BS 控制符的用法。

```
100  REM  BACKSPACE DEMO
110  DIM L$(5),M$(5)
120 B$ =  CHR$ (92): REM  BACK SLASH
130 V$ =  CHR$ (124): REM  VERTICAL BAR
140 BB$ = B$ + "   " + B$
150 VV$ = V$ + V$ + V$ + V$ + V$
170 L$(1) = "00000":M$(1) = "-----"
180 L$(2) = "0   0":M$(2) = "X   X"
190 L$(3) = "0   0":M$(3) = "/   /"
200 L$(4) = "0   0":M$(4) = BB$
210 L$(5) = "00000":M$(5) = VV$
220  REM  CONTROL CODES
230 D$ =  CHR$ (4)
240 BS$ =  CHR$ (8)
250 B5$ = BS$ + BS$ + BS$ + BS$ + BS$
260  PRINT D$;"PR#1"
270  PRINT
```

```
280   PRINT "BACKSPACE DEMO"
290   PRINT
300   FOR I = 1 TO 5
310   PRINT L$(I);B5$;M$(I)
320   NEXT I
330   PRINT D$;"PR#0"
340   END
]RUN
BACKSPACE DEMO

⊖⊖⊖⊖⊖
⊠   ⊠
Ø   Ø
⦸   ⦸
⦶⦶⦶⦶⦶
```

程序中 B $ 是反斜线(＼)，V $ 是垂直线(|)。在每一行上先用 0 或空格印出 5 个符号(第 310 句)，再印出 5 个退格符，于是打印头回到了开始的位置，接着再印出的 M $ (I)字符串就重叠地印在第一次打印的字符上。值行注意的是，连通打印机后(260 句)，最好紧接着再 PRINT 一次，送出一个空行，否则的话下一行的第一个字符常常会丢失。

7.4.2 字型的控制

如果在你打印的文件或报表中，能够控制打印字符的字型和浓度变化，那么打印出的文件一定会更加美观。表7.2 列出了常用字型的控制命令。在这些控制命令中，有些是用 ESC 字符串。例如，ESC+ G 就表示要送出一个 ESC(=CHR $ (27))，然后再送出字符 G。

表 7.2 字符和浓度的控制符

控制功能	控制符号		对应的ASCII 码值	
	开始	结束	开始	结束
标准型字宽	SO	DC4	14	20
缩小型字	SI	DC2	15	18
缩小型宽字	SI+ SO	DC	15, 14	18
每个字印两次	Esc+ G	Esc+ H	27, 71	27, 72
加强型字	Esc+ E	Esc+ F	27, 69	27, 70
加强型印两次	Esc+ E+ Esc+ G	Esc+ H+ Esc+ F	27, 69, 27, 71	27, 72, 27, 70

下面的程序是一个字型控制的示范程序:

```
1000  REM  TEST PRINTING STYLE
1010 D$ =  CHR$ (4)
1015  PRINT D$;"PR#1"
1016  PRINT
1020 SI$ =  CHR$ (15)
1030 SO$ =  CHR$ (14)
1040 ESC$ =  CHR$ (27)
1050 RST$ =  CHR$ (18)
1060 :
1070  REM  DEACTIVATE SCREEN
1080  POKE 1657,80
1090 :
1100  REM  NORMAL TYPE
1110  PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
1120 :
1130  REM  WIDTH PRINTING
1140  PRINT SO$;
1150  PRINT "ABCDEFGHIJKM"
1160 :
1170  REM  COMPRESSED PRINTING
1180  PRINT SI$;
```

```
1190  PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
1200 :
1210  REM  COMPRESSED WIDTH PRINTING
1220  PRINT SO$;
1230  PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
1240  REM  RESET EVERY-THING
1250  PRINT RST$
1260  PRINT D$;"PR#0"
1270  END
]RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKM
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
]LIST
```

下面的程序是一个浓度控制的示范程序:

```
1000  REM  DENSITY DEMO
1010 D$ =  CHR$ (4)
1015  PRINT D$;"PR#1"
1016  PRINT
1020 ESC$ =  CHR$ (27)
1030 DN$ = ESC$ + "G"
1040 DF$ = ESC$ + "H"
1050 EN$ = ESC$ + "E"
1055 EF$ = ESC$ + "F"
1060 :
1070  REM  DEACTIVATE SCREEN
1080  POKE 1657,80
1090 :
1100  REM  NORMAL DENSITY
1110  PRINT "NORMAL DENSITY"
1120 :
1130  REM  DOUBLE STRIKE DENSITY
1140  PRINT DN$;"DOUBLE STRIKE DENSITY";DF$
1160 :
1170  REM  EMPHASIZED DENSITY
1180  PRINT EN$;"EMPHASIZED DENSITY";EF$
1200 :
```

```
1210  REM  DOUBLE+EMPHASIZED DENSITY
1220  PRINT DN$;EN$;"DOUBLE STRIKE AND EMPHASIZED DENSITY";EF$;DF$
1260  PRINT D$;"PR#0"
1270  END
```

NORMAL DENSITY
DOUBLE STRIKE DENSITY
EMPHASIZED DENSITY
DOUBLE STRIKE AND EMPHASIZED DENSITY

7.4.3 行间距控制

通常打印机在标准情况下每一英寸打印六行字符，也就是每1／6英寸一行。我们也可以用控制字符来改变行与行之间的距离，其格式如下：

Esc + A + n　1 ⩽ n ⩽ 85

上式中的n是一个在1到85之间的十进制整数。当把这个控制符送给打印机之后，每次换行时，行与行之间的距离为n／72英寸。使用该控制字符时要注意的是，送出n时，必须要用CHR $ (n)把它转换成对应的ASCII字符。

下面是一个行间距控制的示范程序：

```
]LIST

100  REM  VARIABLE SPACEING DEMO
110 ESC$ =  CHR$ (27)
120 SP$ = ESC$ + "A"
130 D$ =  CHR$ (4)
135 CR$ =  CHR$ (13)
140  PRINT D$;"PR#1"
145  PRINT
150  POKE 1657,80
160  FOR I = 2 TO 18 STEP 2
170  PRINT SP$ +  CHR$ (I);
180  FOR J = 1 TO 3
190  PRINT "LINE SPACING=";I;"/72 INCH"
200  NEXT J
210  PRINT SP$ +  CHR$ (12) + CR$
220  NEXT I
```

```
230  POKE 1657,40
240  PRINT D$;"PR#0"
250  END

]RUN

LINE SPACING=2/72 INCH
LINE SPACING=2/72 INCH

LINE SPACING=4/72 INCH
LINE SPACING=4/72 INCH

LINE SPACING=6/72 INCH
LINE SPACING=6/72 INCH
LINE SPACING=6/72 INCH

LINE SPACING=8/72 INCH
LINE SPACING=8/72 INCH
LINE SPACING=8/72 INCH

LINE SPACING=10/72 INCH
LINE SPACING=10/72 INCH
LINE SPACING=10/72 INCH

LINE SPACING=12/72 INCH
LINE SPACING=12/72 INCH
LINE SPACING=12/72 INCH

LINE SPACING=14/72 INCH
LINE SPACING=14/72 INCH
LINE SPACING=14/72 INCH

LINE SPACING=16/72 INCH
LINE SPACING=16/72 INCH
LINE SPACING=16/72 INCH

LINE SPACING=18/72 INCH
LINE SPACING=18/72 INCH
LINE SPACING=18/72 INCH
```

7.5 汉字状态下使用打印机

汉字系统中,汉字的打印驱动程序是针对 MS- 80 Ⅲ型打印机设计的,并要求打印机的接口为 Centronics 接口,打印机接口套接在 I / O 槽口的 1 号槽上。因此有的打印机不能打印汉字。

在进入汉字系统后,可使用 BASIC 命令来控制和打印汉字。

(1)设置打印方式

格式: POKE 1659, N

说明: N =0 为不打印,以屏幕作为输出显示设备。

N =1 ~ 15 为打印,以打印机作为输出设备,所打印的字型与N 的值有关,系统共提供 15 种字型,N =1 最小,N =15 最大。

N 的值以 16 为模,若 N- INT(N / 16) * 16 =0 则系统退出汉字打印模式,否则打印机被启动为汉字打印模式。

(2)设置字间距

格式: POKE 1687, N

说明: N 为所打印的字符及汉字之间的字间距离,N 的取值为 0 ~ 255 之间,,即字间距离可在 0 ~ 255 个点之间。初次进入打印方式后,字间距自动设置为 1。

(3)设置行距

格式: POKE 1915, N

说明: N 为打印行之间的点间距,N 的范围在 0 ~ 255 之间,即行间距离在 0 ~ 255 个点间距之间,初次进入打印方式后,行间距自动设置为 1。

(4)设置行打印字数

格式: POKE 2043, N

说明: N 为每行所打印的汉字个数。应注意一个汉字占有两个字符的位置。N 取值范围在 0 ~ 255 之间,初次进入打印方式后,自动设置为每行打印 40 个汉字,即 N =。

一行所允许打印的汉字个数与打印机型号有关。

下面是汉字打印简单举例。

```
]LIST

10 A$ = "中华学习机"
20  FOR I = 15 TO 1 STEP  - 1
30  POKE 1659,I
40  PRINT A$
50  NEXT I
55  POKE 1659,5
60  FOR I = 20 TO 1 STEP  - 3
65  POKE 1787,I
70  PRINT A$
80  NEXT I
90  FOR I = 10 TO 1 STEP  - 1
95  POKE 1915,I
100  PRINT A$
110  NEXT I
120  POKE 1659,0

]RUN
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
中华学习机
```

中　　华　　学　　习　　机
中　 华　 学　 习　 机
中　 华　 学　 习　 机
中　华　学　习　机
中 华 学 习 机
中华学习机
中华学习机
中华学习机

中华学习机

中华学习机

中华学习机

中华学习机

中华学习机

中华学习机

中华学习机
中华学习机
中华学习机

7.6　　高分辩率图象的打印

通常与 CEC－Ⅰ中华学习机配备的打印机都具有图形打印的功能，可以把高分辩率图形页的内容在打印机上打印出来。例如我们要打印第一页上的图形，首先应该把图形数据载入高分辩率图形第一页的地址区域（$ 2000 ～ $ 3FFF），然后用 PR#1 命令连通打印机，接下来我们就可发出命令进行图形的打印了。

在立即型方式下，只要按 CTRL－ Q 键，不必按回车，打印机就会立即打印出第一页的图形。

图象打印完以后，最好用 CTRL－ X 命令消除掉 CTRL－ Q 命令，然后再用 PR#0 使打印机脱离开与主机的连通。图7.1 就是用 CTRL－ Q 打印的高分辩率第一页图形的例子。

由于屏幕上点的显示方式刚好与打印机的打印方式相反，即对应于“1”的点在屏幕上显示白点，而在打印机上打印的刚好是黑点。因此图7．1 皆图象与屏幕上所看到的是反相的，就象是照像机的底片样。



图7.1 高分辨图形打印的例子(正向图形)

CEC-Ⅰ中华学习机为用户提供了一个8位的输出寄存器,对应于1号槽上的打印机来说,其内存地址为1913($799)。当用户给这个输出寄存器输入相应的数值控制时,就能控制高分辨率图形页打印的方式。输出寄存器各位码的含义如下:

U	D	I		L		P	
行方式	放大	反相	异或	或	与	第二页	第一页
128	64	32	16	8	4	2	1

当该寄存器中的某一位为1时,便执行相应的操作。

在BASIC语言系统下,控制图形页打印方式命令为:

POKE 1913,N(n =U + D + I + L + P)

命令中各参数的含义如下:

① P参数为高分辨图形页选择。P =1时,打印高分辨图形第一页;P =2时,打印高分辨图形第二页;P =3时,将两页图形并排打印,第一页在左,第二页在右。

② L参数决定二页高分辨图形逻辑操作。L =0,无逻辑操作;L =4,把第一页和第二页图中的点经过逻辑"与(AND)"操作以后打印出来;L =8,把两页图形中的点经过逻辑"或(OR)"操作以后打印出来;L =16,把两页图形中的点经过"异或(EOR)"操作以后打印出来。

③ I参数决定正相或反相打印方式。I =0,表示正相打

印;I =32,表示反相打印。

④ D参数决定打印图形的比例 D =0为正常打印;D =64,把图形放大一倍打印。

⑤ U参数设置行打印方式。U =0,对整幅图象进行打印。U =128,仅打印由VTAB定位的那一行图象。

高分辩图形打印可以用立即型方式也可以用程序方式。在立即型方式下,连通打印机后,按CTRL- Q键,不用按回车,就可以立即打印高分辩图象;在程序方式下执行PRINT CHR $ (17)语句,就可以打印高分辩图象。

下面是一个控制图形页反相打印方式的例子,如图7.2所示。

```
10  REM  HI-GRAPHIC PRINTING DEMO
20  PRINT  CHR$ (4);"BLOAD H10,A$2
     000"
30  POKE 1913,33
40  PRINT  CHR$ (4);"PR#1"
50  PRINT  CHR$ (17)
60  PRINT  CHR$ (24)
70  PRINT  CHR$ (4);"PR#0"
```



图7.2 高分辩图形页反相打印

附录1 编辑命令

在本附录中，将对 CEC- I 中华学习机键盘上的具有编辑功能的各键作一个综合说明。

ESC 键的组合

下面有 11 个编辑命令，是根据 ESC 键和另一个键的组合而成，按键的顺序是：先按下 ESC 键，并放开，然后再按另一个键。

ESC- A：把光标的位置向右移动一格，它既不改变屏幕显示的内容，也不改变内存的内容。

ESC- B：把光标的位置向左移动一格，它既不改变屏幕显示的内容，也不改变内存的内容。

ESC- C：把光标的位置向下移一行，它既不改变屏幕显示的内容，也不改变内存的内容。

ESC- D：把光标的位置向上移动一行，它既不改变屏幕显示的内容，也不改变内存的内容。

ESC- K：与 ESC- A 功能相同，但不退出编辑工作方式，再按一次 ESC 键则可退出编辑状态。

ESC- J：与 ESC- B 功能相同，但不退出编辑工作方式，再按一次 ESC 键则可退出编辑状态。

ESC- M：与 ESC- C 功能相同，但不退出编辑工作方式，再按一次 ESC 键则可退出编辑状态。

ESC- I：与 ESC- D 功能相同，但不退出编辑状态，再按一次 ESC 键则可退出编辑状态。

ESC- E：删去从光标所在位置开始，到该显示行的末尾之间的所有字符。

ESC- F：删去光标所在位置开始，到显示屏幕上的最后一个字符为止的全部字符。

ESC- @：把显示屏幕全部清除干净，并将光标移到屏幕的左上角。

键：与 ESC- I 功能相同，但不用按 ESC 键。

键：与 ESC- M 的功能相同，但不用按 ESC 键。

> 键:使光标沿着显示行向前移动,它经过的每一个字符都会重新复制到内存中,就象在在键盘上重新按键输入一样。但是该键并不改变屏幕显示的内容。

< 键:使光标沿着显示行向后退一格,它经过的每一个字符都从内存中抹去。但是该键并不改变屏幕显示的内容。

CTRL- X:将屏幕现行显示行的字符全部删去,并将光标移到下一行的最左边。

CTRL- C 与 Quit:能使正在执行中的 BASIC 程序中断。

CORL- B:清除内存中的 BASIC 程序及所有变量。

附录2　CEC-BASIC命令参考

(1)　简单变量

类型	名	范围
实型	AB	+/- 9.99999999E+37
整型	AB%	+/- 32767
串	AB $	0到255个字符

这里A是字母,B可以是字母也可以是数字。名可以由二个以上的字符组成,但仅有最前面的二个字符是有意义的:AB%和AB3QS%是同一整数变量。

(2)数组变量

类型	典型的元素名
实型	AB(3,12,7)
整型	AB%(3,12,7)
串	AB $ (3,12,7)

数组的大小是受有效内存所限制的。

(3)代数运算符

=	赋值给变量(LET是选用项)
—	负号
∧	次幂
*	乘
/	除
+	加
—	减

(4)关系和逻辑运算

=	等于
< >	不等于
<	小于

>	大于
< =	小于或等于
> =	大于或等于
NOT	逻辑“非”
AND	逻辑“与”
OR	逻辑“或”

关系和逻辑表达式为真时，其值为 1，为假时，其值为 0，逻辑运算符也能够用于串的比较。

(5)系统和实用命令

LOAD	从磁带上装入程序。
SAVE	把程序存入磁带的。
NEW	删除当前程序。
RUN	从最小行号处开始运行程序。
RUN X	从 X 行开始运行程序。
STOP	停止执行并报告停止在哪一行。
END	停止执行，但不显示信息。
CTRL- C	用于立即执行方式来停止程序或列表。
RESET	无条件地转向监控程序。用 CTRL-C 或 OG 来返回到 CEC-BASIC。
CONT	继续执行由 STOP，END 或 CTRL-C 所终止的程序。
TRACE	提供调试手段，列出它每次新执行列的每一行的行号。
NOTRACE	关闭 TRACE。
PEEK(X)	送回存储单元 X 的内容。
POKE X,N	将内存单元 X 的内容修改成 N。
WAIT X,Y,Z	等待，直到 X 单元的内容与 Z 进行 XOR 运算，并且再与 Y 进行 AND 运算所得列的值为非零时为止。
CALL X	转向以内存单元 X 开始的机器语言子程序。
PLAY	装入并执行磁带游戏软件。

MUSIC X,Y	音乐语句,X 为频率,Y 为时间。
LG	装入并执行固化的 LOGO 语言。
HIMEM:	置 CEC-BASIC 程序可使用的最高内存有效地址。
LOMEM:	置 CEC-BASIC 程序可使用的最低内存有效地址。

(6)编辑命令和与格式有关命令

LIST	列出整个程序。
LIST X-Y	列出从 X 行到 Y 行之间的程序段。
REM XYZ	用来写出程序的注解,程序将这部分忽略。
VTAB Y	把光标移到 Y 行(1 到 24)。
HTAB X	把光标移到 X 位置上(1 到 40)。
TAB(X)	仅用于 PRINT 语句,它把光标移到 X 位置上(1 到 40)
POS(0)	送回光标当前所在水平位置的值(0 到 39)。
SPC(X)	仅用于 PRINT 语句,在最后一次打印的项和要打印的下一项之间放上 X 个空格。
HOME	清除屏幕并把光标放在项部。
CLEAR	所有的变量重置为零。
FRE(0)	送回用户还可使用的内存总数。
FLASH	置计算机的输出为闪烁方式。
INVERS	置计算机的输出为白底里字方式。
NORMAL	关闭闪烁或白底黑字方式。
SPEED=X	置字符输出速率为 X(0 到 255)。
ESC A	把光标向右移一格。
ESC B	把光标向左移一格。
ESC C	把光标向下移一格。
ESC D	把光标向上移一格。
右箭头(→)	把光标下的字符输入内存,并把光标向右移一格。
左箭头(←)	删除当前行中刚刚打入的字符,并把光标向左移一格。
CTRL-X	删除当前打入的行。

(7)数组和串

DIM A(X,Y,Z)　置 A的最大下标，保留X+1*Y+1*Z+1个实型元素的存储空间。由A(0,0,0)元素开始。

DIM A $(X,Y)　置 A $ 的最大下标，它可以包含X+1*Y+1个串元素，每个元素最多可达255个字符。

LEN(A $)　送回A $中所含的字符个数。

STR $(X)　将X的数值转换或对应的字符串送回。

VAL (A $)　以A $中的第一个非数字字符为界，将其前面的字符转换成数值，送回。

CHR $(X)　送回代码为X的ASCII字符。

ASC(A $)　送回A $中的第一个字符的ASCII代码。

LEFT $(A $,X)　送回A $中的最左边的X个字符。

RIGHT $(A $,X)　送回A $中的最右边的X个字符。

MID $(A $,X,Y)　送回A $中的从字符X开始的Y个字符。

+　用于连接串的运算符。

STORE A　把数值数组A保存到磁带上，不可直接用来存储串数组。

RECALL B　把数组从磁带上装回，B数组必须已经被正确地用DIM定义过。

(8) 输入 / 输出命令

(同时参考LOAD和SAVE,SAVE,STORE和RECALL)。

INPUT A $　在屏幕上显示一个问号(?)等待用户打入一组字符串值给A $。

INPUT "XYZ";A　在屏幕上打印出XYZ,等待用户打入一个实数值给A

GET A $　等待用户打入单个字符值给A $,它无需按回车键。

DATA X,"Y",Z　建立READ语句能使用的数据元素表。

READ A $ 将下一个 DATA 的元素赋给 A $。

RESTORE 重新从第一个 DATA 元素开始，用 READ 来读。

PRINT "X =";X 在屏幕上打印串 X = 和变量 X 的值，分号表示紧接着上一项段打印，逗号则表示各打印项之间留有三个制表域的长度。符号？也表示 PRINT 语句。

IN# X 从第 X 插口上的外部设备上进行预输入。

PR# X 置输出为第 X 插口上的外部设备。

LET X=Y 把 Y 的值赋给变量 X，LET 是选用项。

DEF FN A(X) =X+ 23 / X

定义函数 FNA，在后面使用时，FNA 的变量由定义中的表达式 X 来代替。FNA(4)将送回值 9.75。

(9)有关流程控制的命令

GOTO 347 转向 347 行。

IF X=3 THEN STOP

如果断言 X=3 为真（非零），则继续执行 THEN 后面的语句。如果断言为假则执行下一行号的语句

FOR X=1 TO 20 STEP 4...NEXT X

执行 FOR 语句和相应的 NEXT 语句之间的所有语句，首先 X=1，然后 X=5，X=9....。直到 X〉20 才继续执行 NEXT 语句后面的程序。如果末指明 STEP 的大小，则表明 STEP 大小于 1

NEXT X 定义 FOR...NEXT 循环的底，X 是选用项。

GOSUB 33 转向行号为 33 开始的子程序。

RETURN 标明子程序的结束，返回到最后的 GOUSUB 的下一条语句处执行。

POP 从 RETURN 地址栈中移出一个地址。

ON X GOTO 397,12,458

转向表中所列出的第 X 个行号所指出的语句。如果 X=2,则转向 12 行。

ON X GOSUB 397,12,458

转向表中第 X 个行号的子和序。

ONERR GOTO 4500

在后继语句中,如产生错误使得程序转到错误处理程序(在 4500 行)上执行,而不是显示信息,也不停止程序的运行。

RESUME　在错误处理程序中,使得返回发生错误的语句处执行。

(10) 作图和游戏控制

①低分辨率图形

GR　置成低分辨率作图方式,把项端的 40 × 40 区域清成黑色,底部留有 4 行正文显示。

COLOR =X　为画下一个点置颜色(0 到 15)。

PLOT X,Y　把有色的点显示在横坐标为 X,纵坐标为 Y 的位置上,X 和 Y 在 0 到 39 范围内(0,0)是屏幕的左上角。

HLIN X1,X2 AT Y

从点(X1,Y)到点(X2,Y)画一条水平线。

VLIN Y1,Y2 AT X

从点(X,Y1)到点(X,Y2)画一条直线。

SCRN(X,Y)　送回在屏幕上(X,Y)这一点的颜色数。

②高分辩图形

HGR　置高分辩率作图方式(第 1 页),把项端的 280 × 160 区域清成黑色,底部留出 4 行的正文显示。

HGR2　置第 2 页的高分辩作图方式,按整个

280 × 192 区域的屏幕清成黑色。

HCOLOR=X　为画下一个点置颜色(0 列 7)

HPLOT X,Y　把有色点置在横坐标为 X,纵坐标为 Y 的位置上,X 的变化范围从 0 到 279,Y 则从 0 到 159(对 HGR),或到 191(HGR2)。(0,0)是左上角。

HPLOT X1,Y1 TO X2,Y2

从点 X1,Y1 到点 X2,Y2 画一条线,命令还可以扩充,加上 TO Xn,Yn。

SHLOAD　从磁带上装入一个图形表。

DRAW 3 AT X,Y

按照图形表中所确定的第 3 号形状来绘制图形,所画的每一点的颜色数是屏幕上那一点原有的颜色数的补数。

POT=X　为执行 DRAW 或 XDRAW,可将形状进行旋转,ROT=0 表示垂直,ROT=16 表示顺时针方向右旋 90 度,ROT=32 表示顺时针方向右旋 180 度等等。

SCALE=X　置 DRAW 或 XDRAW 图形的比例因子(1 到 255)。

③游戏控制

PDL(X)　送回游戏控制器 X(0 到 3)所置的值,它可以在 0 到 255 之间变化。

PEEK(X−16287)

如果其值>127,表示游戏控制器 X(0 到 2)上的按钮已被按下。

PEEK(−16336)　使喇叭发出卡哒声。

(11)一些常用数学函数

SIN(X)　送回弧度 X 的正弦值。

COS(X)　送回弧度 X 的余弦值。

TAN(X)　送回弧度 X 的正切值。

ATN(X)　送回 X 的反正切值，以弧度为单位。

INT(X)　送回小于等于 X 的最大整数。

RND(1)　每次使用时，总送回一个 0 到 0.999999999 的随机数。

RND(0)　再次送回上一次的随机数。

RND(−3)　送回的值为 4.48217178E−08，对于每一个不同的负自变量，总是送回相应于不同变量的一个固定值(换言之，一个函数产生的随机数是一定值，与调用次数无关)。使用了这一随机数后，带有正自变量的 RND 的函数，也产生一固定序列。

SGN(X)　如果 X〈0，送回值为 −1，如果 X=0，送回值为 0，如果 X〉0，送回值为 1。

ABS(X)　送回 X 的绝对值。

EXP(X)　送回 E 的 X 次方的值。e=2.718289。

LOG(X)　送回 X 的自然对数的值。

SQR(X)　送回 X 的正平方根。

附录3 CEC-BASIC零页的使用

单元位置(16 进制)	说明
$ 00 ~ $ 05	继续 CEC-BASIC 的转移指令。
$ 0A ~ $ 0C	USR 函数的转移指令地址，见 USR 函数描述。
$ 0D ~ $ 17	通常用于 CEC-BASIC 的计数器 1 标志。
$ 20 ~ $ 23	分别用来设定屏幕显示的上，下，左、右起始及终止位置或宽度。
$ 24 ~ $ 25	确定字符在屏幕上显示的位置。
$ 2C ~ $ 2D	内含值分别为 HLIN 和 VLIN 画直线时，直线最右和最底端点座标。
$ 30	设定低分辨作图点的颜色。
$ 32	屏幕显示字符的性质标志(正常、反相、闪烁)。
$ 50 ~ $ 61	通常用作 CEC-BASIC 的指针。
$ 62 ~ $ 68	上次乘法 / 除法的结果。
$ 67 ~ $ 68	指向程序开始的指针，对 ROM 版本来讲通常置成 $ 801。
$ 69 ~ $ 6A	指向简单变量区域开始位置的指针，并指向程序的结束，并加上 1 或加上 2，除非用 LOMEM: 语句来修改。
$ 6B ~ $ 6C	指向数组区域开始位置的指针。
$ 6D ~ $ 6E	指向使用中的数值区域的末端的指针。
$ 6F ~ $ 70	指向串存储区域开始的指针，串可从这里开始一直存储到内存的末端。
$ 71 ~ $ 72	通用指针。
$ 73 ~ $ 74	CEC-BASIC 可用的内存最高单元加 1。在最初进入 BASIC 时，置成最高的可用的 RAM 内存地址。
$ 75 ~ $ 76	当前正在执行的程序行的行号。
$ 77 ~ $ 78	“旧行号”通过 CTRL-C，STOP 或 END 语句来建立的，给出程序执行时被中断的行号。

$ 79～$ 7A	“旧的正文指针”指出下一次将要执行的语句在存储器中的地址。
$ 7B～$ 7C	当前 DATA 语句的行号，该行中的数据正由 READ 语句来读。
$ 7D～$ 7E	指出内存中的绝对单元地址，READ 语句正在从这个单元地址中 DATA 中的数据。
$ 7F～$ 80	指向当前输入的源程序的指针，在 INPUT 语句执行期间置为 $ 201，在一条 READ 语句执行期间被置成在程序中 DATA 是从中读入的单元。
$ 81～$ 82	保存上次使用的变量名。
$ 83～$ 84	指向上次使用的变量的值的指针。
$ 85～$ 9C	通用。
$ 90～$ A3	主要的浮点累加器。
$ A4	通常用于浮点的数学例行程序。
$ A5～$ AB	次要的浮点累加器。
$ AC～$ AE	通常用于标志 / 指针。
$ AF～$ B0	指向程序的末尾(不能用 LOMEM: 改变)。
$ B1～$ C8	CHRGEJ 例行程序。CEE-BASIC 每次希望得到另一个字符时，在这里调用它。
$ B8～$ B9	指向通过 CHRGET 例行程序得到的上一个字符的指针。
$ C9～$ CD	随机数。
$ D0～$ D5	高分辩率制图划线指针。
$ D8～$ DF	ONERR 指针 / 划线。
$ E0～$ E2	高分辩制图的 X 和 Y 座标。
$ E4	高分辩制图的颜色字节。
$ E5～$ E7	通常用于高分辩率制图。
$ E8～$ E9	指向图形表开头的指针。
$ EA	高分辩率制图的碰撞计数器。
$ F0～$ F3	通用标志。
$ F4～$ F8	NERR 指针。

附录4　汉字系统的实现方法及内部子程序调用

(1)汉字系统的实现方法

CEC－Ⅰ中华学习机利用了主机硬件所提供的辅助存储器来实现汉字系统的扩充。其汉字字库、汉字管理程序均不占用用户的程序区，它与用户程序之间通过I／O扩充槽口的＄300～＄C3FF进行连接、实现汉字的输入、显示和打印等等。

我们知道，在监控管理程序中，I／O操作是通过监控程序的KSW(＄38，＄39)和CSW(＄36，＄37)与键盘输入和屏幕输出的管理程序联系在一起的，通过KSW，CSW进入并执行的。因此要进入汉字系统就是要将这二个跳转向量指向汉字的输入和输出管理程序。

在没有装入DOS操作系统时，KSW和CSW分别指向＄FD1B和＄FDF0。进入汉字系统后，KSW和CSW则分别指向汉字的输入／输出程序入口＄C303和＄C32B

(2)内部子程序调用

在汉字管理程序中，有许多子程序，用户可以通过直接调用的方法来实现汉字的输入输出，以及扩充汉字的输入方法等。这里仅说明在＄C300～＄C3FF这一段子程序入口地址及使用方法。

CSWA (＄C32B)

完成ASCII字符及汉字的输出，它要求在A寄存器中存放的是字符或汉字的显示码及控制命令码。对于汉字必须采用三字节的机内码，需要分三次调用才能输出一汉字。

GB.CSWA (＄C322)

完成的功能和CSWA完全一样，但它要求汉字代码采用二字节，最高位为1的国标码，需要分二次调用才能输出一汉字。ASCII字符显示码最高位必须为0。

GETLN (＄FD6F)

这是监控程序中的行输入子程序，它将通过KSW间接地调用汉

字的 KSWA($ C303),是用户得到键入汉字的程序入口。

ZT.XSI ($ C36E)

调用汉字系统的显示状态字子程序。将系统当前的状态显示到状态行上。

ZT.XS2($ C377)　　显示键入提示字符。

ZT.XS3($ C380)　　显示汉字揭示。

USR.DCOD($ C389)　　将异形国标码(即国标码最高位为 1)转换为学习机内码。入口,出口参数都放在 A 寄存器中。

USR.ECOD($ C392)　　将学习机内码转换为异形国标码。入口,出口参数都在 A 寄存器中。

BACKSP($ C39B)　　删除光标处字符,并使光标退一格。

SLECTA1($ C3A4)　　选择辅存 1。

SLECTA2($ C3AB)　　选择辅存 2。

SLECTM1($ C3B2)　　选择主存 1。

SLECTM2($ C3B9)　　选择主存 2。

(3)汉字系统输出字符控制命令

CEC-I 中华学习机除了提供键盘方式的屏幕编辑命令外,在汉字显示方式下,系统还提供了一组以程序方式对屏幕进行编辑的命令和一些输出控制字符,它们是:

CHR $(7)　扬声器发出“嘟”的一声。

CHR $(8)　光标退一格。

CHR $(11)　从光标清至页末。

CHR $(12)　清屏幕,光标为(0,0)。

CHR $(13)　输出回车符,光标移至下一行。

CHR $(14)　置显示方式为正常方式。

CHR $(15)　置显示方式为反相方式。

CHR $(17)　退出汉字系统。

CHR $(18)　显示或清除状态提示字符。

CHR $(19)　暂停输出,按任一键恢复输出。

CHR $(26)　从光标清至行末。

附录5 区位、国际和汉字内码对照表

进入汉字系统后，键入的 ASCII 字符占用一个字节，其代码为 $ 01 ~ $ 7D

键入的汉字均以等长的 3 字节码保存在主机的内存中。其格式为: 7F+ 区码 + 位码。这里的区位码不能完全和国标中的代码一样。这里的区位码是经过转换的，它与国标中规定的区位码如表所示。

例如，汉字“啊”的国标码为 1601。它在机内的表示形式为: 7F 2E 1D。在 BASIC 程序中用字符串函数表示则为:

A $ =CHR $ (127)+ CHR $ (46)+ CHR $ (29)

区位码、国标码、内码对照表

区位码	国标码	异形国标码	学习机内码		区位码	国标码	异形国标码	学习机内码	
10 进制	16 进制	16 进制	16 进制	10 进制	10 进制	16 进制	16 进制	16 进制	10 进制
1	21	A1	1D	29	11	2B	AB	28	40
2	22	A2	1E	30	12	2C	AC	29	41
3	23	A3	1F	31	13	2D	AD	2A	42
4	24	A4	20	32	14	2E	AE	2B	43
5	25	A5	21	33	15	2F	AF	2D	45
6	26	A6	23	35	16	30	B0	2E	46
7	27	A7	24	36	17	31	B1	2F	47
8	28	A8	25	37	18	32	B2	30	48
9	29	A9	26	38	19	33	B3	31	49
10	2A	AA	27	39	20	34	B4	32	50

区位码	国标码	异形国标码	学习机内码		区位码	国标码	异形国标码	学习机内码	
10 进制	16 进制	16 进制	16 进制	10 进制	10 进制	16 进制	16 进制	16 进制	10 进制
21	35	B5	33	51	54	56	D6	55	85
22	36	B6	34	52	55	57	D7	56	86
23	37	B7	35	53	56	58	D8	57	87
24	38	B8	36	54	57	59	D9	58	88
25	39	B9	37	55	58	5A	DA	59	89
26	3A	BA	38	56	59	5B	DB	5A	90
27	3B	BB	39	57	60	5C	DC	5B	91
28	3C	BC	3B	59	61	5D	DD	5C	92
29	3D	BD	3C	60	62	5E	DE	5D	93
30	3E	BE	3D	61	63	5F	DF	5E	94
31	3F	BF	3E	62	64	60	E0	5F	95
32	40	C0	3F	63	65	61	E1	60	96
33	41	C1	40	64	66	62	E2	61	97
34	42	C2	41	65	67	63	E3	62	98
35	43	C3	42	66	68	64	E4	63	99
36	44	C4	43	67	69	65	E5	64	100
37	45	C5	44	68	70	66	E6	65	101
38	46	C6	45	69	71	67	E7	66	102
39	47	C7	46	70	72	68	E8	67	103
40	48	C8	47	71	73	69	E9	68	104
41	49	C9	48	72	74	6A	EA	69	105
42	4A	CA	49	73	75	6B	EB	6A	106
43	4B	CB	4A	74	76	6C	EC	6B	107
44	4C	CC	4B	75	77	6D	ED	6C	108
45	4D	CD	4C	76	78	6E	EE	6D	109
46	4E	CE	4D	77	79	6F	EF	6E	110
47	4F	CF	4E	78	80	70	F0	6F	111
48	50	D0	4F	79	81	71	F1	70	112
49	51	D1	50	80	82	72	F2	71	113
50	52	D2	51	81	83	73	F3	72	114
51	53	D3	52	82	84	74	F4	73	115
52	54	D4	53	83	85	75	F5	74	116
53	55	D5	54	84	86	76	F6	75	117

区位码	国标码	异形国标码	学习机内码		区位码	国标码	异形国标码	学习机内码	
10 进制	16 进制	16 进制	16 进制	10 进制	10 进制	16 进制	16 进制	16 进制	10 进制
87	77	F7	76	118	91	7B	FB	7A	122
88	78	F8	77	119	92	7C	FC	7B	123
89	79	F9	78	120	93	7D	FD	7C	124
90	7A	FA	79	121	94	7E	FE	7D	125

附录6 DOS3.3命令参考

(1)磁盘管理命令

INIT X	初始化一个空白磁盘,使之成为工作盘,分配给欢迎程序的名为X。
CATALOG	显示卷号和所有在这个磁盘上文件的名称,这些文件的有关信息,如区段数和类型也显示出来。
SAVE X	保存文件名为X的文件到磁盘上。不发任何警告,就改写了以前同类、同名的文件。
LOAD X	在清除存储器之后,把BASIC程序文件X装入内存。
RUN X	先LOAD X,然后再运行这个程序。X为BASIC程序文件。
RENAME X,Y	改变磁盘上的文件名X,使之成为新名Y。
DELETE X	从磁盘上抹去文件X。
LOCK X	封锁文件X,以防意外被修改或删除。封锁过的文件在CATALOG时用 * 注明。
UNLOCK X	解除先前被封锁的文件X,以便允许更改或删除文件X。
VERIFY X	检查文件X的内部一致性。如果X的保存没有错误,则无信息给出。

MON 〔,C〕〔,I〕〔,O〕

引起了磁盘命令(C),来自磁盘的输入信息(I),向磁盘输出的信息(O)在屏幕上显示。无参数时,该命令被忽略。

NOMON 〔,C〕〔,I〕〔,O〕

撤消由MON引起的显示。当无参数时,该命令被忽略。

MAXFILES n　对于磁盘的输入 / 输出保留几个文件的缓冲区(引导时,保留3个文件缓冲区)。此命令在

LOAD 或 RUN 一个程序之前使用。

(2)顺序文件命令

OPEN X	打开或建立顺序文本文件 X,并分配一个文件缓冲区,准备从文件的开头开始进行读/写。
CLOSE〔X〕	中止写 X 的工作,并撤消分配给文本文件 X 的文件缓冲区。不用文件名 X 时,文件(除一个 EXEC 文件外)全部关闭。
WRITE X〔,Bb〕	连续的 PRINT 把字符发送给正文文件 X,此 X 是顺序文本文件。在当前文件位置(如果有说明的话)的开头或在字节 b 上进行写操作。任何 DOS 命令都可消除该命令的作用。
READ X〔,Bb〕	连续的 INPUT 和 GET 从顺序文件那里取得字符。在当前文件位置的开头(如果有说明的话)或在字节 b 上进行读操作。INPUT 所对应的是一个记录。任何 DOS 命令都可消除该命令的作用。
APPEND X	类似于 OPEN,打开所存在的顺序文件 X,但是它准备在这文件的末尾进行写操作。
POSTION X,Rp	在打开的顺序文件 X 里,从当前位置开始向前移动 p 个记录作为当前位置。该命令取消 WRITE 和 READ 命令的作用。
EXEC X〔,Rp〕	执行顺序文本文件中连贯的记录就好象它们是从键盘上打入的一样。使用 Rp 参数,则在第 p 个记录上开始执行,记录可以包括 BASIC 程序行和立即执行的 BASIC 命令或 DOS 命令。

(3)随机文件命令

OPEN X,Lj	打开或建立随机文本文件 X,分配一个文件缓

冲区，定义的记录长度为 j 个字节，准备从记录 0 的始端进行 WRITE 或 READ。每次当文件被打开时，必须使用同样的长度参数。

CLOSE 〔X〕　中止写文件 X 的工作，并撤消分配给该文件的缓冲区。如果不包含文件名参数，则该命令关闭所有已打开的文件。

WRITE X 〔,Rr〕〔,Bb〕

连续发生的 PRINT 将字符发送到随机文件 X 里。如果不用参数，则从当前文件位置开始进行写操作。如果单独用了 Rr 参数，则该命令在记录 r 的字节 0 上开始执行。若使用 Bb 参数，则写操作在当前或被说明的记录上第 b 个字节开始。任何 DOS 命令都能取消该命令的作用。

READ X〔,Rr〕〔Bb〕

连续的 INPUT 或 GET 从随机文件 X 中取得所要的字符。参数的使用与 WRITE 命令相同。

(4)机器语言文件命令

BSAVE X,Aa,Lj　在磁盘上保存内存中以地址 a 开头，j 个字节长度的内容。

BLOAD X〔,Aa〕　把二进制文件 X 装入到该文件原来 BSAVE 时的那个存储区域里，或装入到地址 a 开头的存储区域里。

BRUN X〔,Aa〕　光 BLOAD 二进制文件 X，然后跳转到那个被装入文件的第一个存储地址上运行程序。

附录7 PEEK、POKE和 CALL的用法

这是一些可以借助于 PEEK, POKE 或 CALL 的命令来使用 BASIC 的特殊性能。注意:其中某些与 BASIC 中其它一些命令有完全相同的效果。

简单的开关动作是依赖于地址的,任何包含那个地址的命令对那个开关将起作用。例如:POKE−16304,0 就是一个例子,对于这个例子,你可以通过使用 POKE 来把从 0 到 255 中的任何数放到那个地址中,也可通过 PEEK 来对那个地址进行读取:X=PEEK(−16304)来获得相同的效果。

这对某些命令不适用。在这些命令中,你必须使用 POKE 来把一个特定的值放到所需的地址中,这些特定的值可以置一个页边为空白,或者用来把光标移到一个指定的地方。

(1)置文本显示窗口

在下面例子中行号 10,20,30 和 40 的前四个 POKE 命令是用于置 TV 屏幕上的“窗口”大小。在这个窗口上,显示正文,并且能够自动“上滚”。这些命令分别地置窗口的左边空白,行的宽度。窗口顶部空白和底部空白。

10　POKE　32,L

置左边的空白,其大小由 L 指出的值确定,值的范围在 0 到 39 之间,这儿的 0 是指最左边的位置。

窗口的宽度不是由这条命令来改变的,这意味着右边的空白是根据你移动左边空白的同样数目来变化。为了保护你的程序及 BASIC,首先应适当压缩窗口的宽度,然后改换左边空白。

20　POKE 33,W

TV 显示的宽度(每行字符的个数)为 W 指出的值,这个值在 1 到 40 的范围内。

不能置 W 为零:POKE 33,0 会破坏 BASIC。

如果 W 小于 33,PRINT 命令的第三个制表(TAB)域可能把字符

打印在窗口外。

30　POKE 34,T

根据 T 指定的值建立 TV 显示的顶部空白，其 T 的范围在 0 到 23 之间，这里的 0 是屏幕的顶行。POKE 34,4 将不允许在屏幕的前四行上打印文本。所置的窗口的顶部的空白(T)不能低于底部的空白(下面的 B)。

40　POKE 35,B

根据 B 指定的值建立屏幕底部空白，其 B 的范围是 0 到 24 之间。这儿的 24 是屏幕底部的一行，窗口底部的空白(B)不能高于顶部空白(即上面的 T)。

(2)有关文本显示、文本显示窗口和键盘的一些命令

45　CALL-936

清除文本显示窗口内的所有字符，且移动光标到窗口的左上角。

50　CALL-958

清除在文本窗口中从当前光标位置底部空白处的所有字符，在当前行光标上面的字符以及光标左边的字符将不受影响。

60　CALL-868

清除当前行从光标位置到右边空白。

70　CALL-922

发出一个换行。

80　CALL-912

上滚正文一行，即在所定义的窗口内的每行正文都向上移动一行位置，原来的顶端的行丢失。原来的第二行为第一行。底端的一行现在是空白。在所定义的窗口外的字符不受影响。

90　X=PEEK (-16384)

读键盘，如果 X >127 那么表示已按了一个键模 X 的内容是所按键的 ASCII 码值，且第 7 位为 1。这在长程序中是有用的。在这个程序中，计算机检查用户是否想要用新数据去中断一下程序，而不是停止程序的执行。

100　POKE-16368,0

复位键盘选通，以便下一个字符可以读入，这在读入键盘后应立即去做。

(3)与光标有关的命令

110　CH=PEEK(36)

读回当前光标的水平位置，并且把这个值赋给变量 CH。光标将在 0 到 39 范围内，且是与文本显示窗口的左边空白有关的光标的位置。这个左边空白是由 POKE32，L 置的，因此，如果由 POKE32，5 置了左边空白，那么，窗口中最左边的字符是在从屏幕的左端起的第六个打印位置上，且如果 PEEK(36)送回的值是 5，那么，光标是在从屏幕的左边缘起第十一个打印位置上，即在从文本显示窗口的左起第六个打印位置上(最初听起来有些混乱，因为最左边的位置是 0，而不是 1)。这与 POS(X)函数是相同的(参看下一个例子)。

120　POKE36，CH

移动光标到从文本显示窗口的左起第 CH+ 1 个打印位置上(例：POKE36，0 将使得下一个字符从窗口的左边打印)。如果窗口左边空白在用 PRINT 打印之前必须改变，CH 必须小于或等于由 POKE 22，W 置的窗口的宽度，且必须大于或等于 0。象 HTAB 一样，这命令能够把光标移出正文窗口的右边空白，但仅够打印一个字符。

130　CV=PEEK(37)

读出当前光标的垂直位置，且置 CV 等于该值，CV 是光标的绝对垂直位置，它与正文屏幕的顶部或底部的空白无关。因此，CV=0 是屏幕的顶端行，CV=23 是底行。

140　POKE 37，CV

移动光标到由 CV 指定的绝对垂直位置上。0 是最上面的一行，23 是底行。

(4)有关作图的命令

为了显示文本和制图，中华学习机的内存划分出四个区域：文本显示的第 1 页和第 2 页以及高分辩率图形显示的第 1 页和第 2 页。

①文本显示的第 1 页是所有的文本和低分辩率制图的通用内存区域，这通过 TEXT 和 GR 命令来使用。

②在内存中文本显示的第 2 页刚好位于文本显示的第 1 页的上面。对用户来讲，这页是不容易存取到的。同正文第 1 页一样，存储在正文第 2 页上的信息可以解释为正文，也可以解释为低分辩率图形，或者两者皆可。

③高分辩率图形的第一页驻留在中华学习机的内存的 8K 到 16K 上，这个区域是通过 HGR 命令来使用的。

④高分辩率图形的第 2 页驻留在中华学习机内存的 16K 到 24K 上。这个区域是通过 HGR2 命令使用的。

你可以用 BASIC 的正文和制图命令或者用这四种不同的开关来使用不同的制图和正文方式。如同这里讨论的许多开关一样。PEEK 或 POKE 可对一个地址置开关为一种状态，以及 PEEK 和 POKE 可对第 2 个地址置开关为另一个状态，简单地说，这四种开关的选择区间为:

i 文本显示 (POKE−16303,0)

高分辩率或低分辩率图形显示 (POKE−16304,0)

ii 文本或高分辩率显示的第一页 (POKE−16300,0)

文本或高分辩第显示的第二页 (POKE−16299,0)

iii 文本显示的第一页或制图的第二页 (POKE−16293,0)

制图的高分辩率的第一页或第二页 (POKE−16297,0)

iv 满屏幕高分辩率或低分辩率制图 (POKE−16302,0)

高分辩率或低分辩率制图加文本显示混合方式(POKE−16301,0)

150 POKE−16304,0

从文本显示方式转变到彩色制图方式中，而不清除制图屏幕为黑色。根据另外三个开关的置位，转变成制图方式，可以是低分辩率或高分辩率(从第一页到第二页)，也可以是制图加文本显示的混合方式或满屏幕制图方式。

BASIC 中类似的命令: GR 命令用来转变成第一页的低分辨率制图加正文的混合屏幕，并且清除制图屏幕为黑色;HGR 命令用来转变成为第一页高分辨率制图加文本显示的混合屏幕，并且清图形屏幕为黑色;HGR2 命令用来转变成第二页高分辨率满屏图形且清除整个屏幕为黑色。

160　POKE －16303,0

从任何彩色制图显示方式转变成所有文本显示方式，而不重置“上滚”窗口，根据第一页 / 第二页开关的置位，可将文本显示页从第一页转换成第二页，或从第二页转换成第一页。

TEXT 命令将屏幕置成全文本显示方式，它除了选择文本显示的第一页以外，还重新将窗口复位成最大，并将光标移到显示屏幕的左下角上。

170　POKE －16302,0

转换制图加文本显示的混合屏幕为满屏幕制图显示方式。

根据其它开关的置位，可以作为正文或是一个 40 乘 48 的栅格上的低分辨率制图，或一个 278 乘 192 栅格的高分辨率制图出现。

180　POKE －16301,0

从满屏图形转换成图形加文本显示的混合屏幕方式，在屏幕的底部有 4 行文本显示区，每行 40 个字符。

根据其它开关的置位上，屏幕的上部分可以显示文本，也可以在一个 40 乘 48 的栅格上进行低分辨率制图或在一个 278 乘 160 栅格上进行高分辨率制图。屏幕的两部分显示将来自同样的页数(1 或 2)。

184　POKE －16300,0

从第二页转换到第一页，不清除屏幕或移动光标。当你从 BASIC 进入到整型 BASIC 时这是必须的，否则，你看到的将仍然是内存的第二页的内容。

根据其它开关的置位，这可以使得显示从高分辨率制图的第二页改变到高分辨率制图的第一页，从低分辨率制图的第二页转变为低分辨率制图的第一页，或者从文本显示的第二页转变成文本显示的第一页。

186　POKE 16299,0

从第一页转换到第二页，不清除屏幕或移动光标。

根据其它开关的置位，这可以使得显示从高分辨率制图的第一页改变成高分辨率制图的第二页；从低分辨率制图的第一页改变成低分辨率制图的第二页，或者从正文第一页改变成正文第二页。

190　POKE －16298,0

改变制图页，从一个高分辨率制图的页转变到正文的同样页上，不

清除屏幕。当你从 BASIC 进入到整型 BASIC 时,这是必须的。否则,整型 BASIC 的 GR 指令可能错误地显示高分辨率页。

依赖于其它开关的置位,这可以使得显示从高分辨率制图的第一页改变为低分辨率制图的第一页;从高分辨率制图的第二页改变到低分辨率制图的第二页;或者(在正文方式下)可以使得显示不改变。

195　POKE －16297,0

对图形换页,从一个文本显示页到高分辨率的相应页,不清除屏幕。

依赖于其它开关的置位,这可以使得显示从低分辨率图形第一页变换到高分辨率图形的第一页,从低分辨率图形的第二页变换到高分辨率图形的第二页,或者(在正文方式下)可以使显示不改变。

200　CALL－1994

清除文本显示的第一页的上面 20 行为保留的符号 @,如果你是在第一页低分辨率制图方式中,则清除制图屏幕的上面 40 行为黑色,这对正文第二页或对高分辨制图不起作用。

205　CALL－1998

清除整个文本显示的第一页为保留的符号 @。如果你是在第一页低分辨率满屏制图方式中,则清除整个屏幕为黑色。这对正文第二页或对高分辨率制图不起作用。

200　CALL　62450

清除当前高分辨率屏幕为黑(BASIC 记住上次使用的屏幕,而不考虑开关的置位)。

210　CALL　62454

清除当前高分辨率屏幕为最近绘制使用的高分辨率的颜色(HCOLOR)(BASIC 记住最后使用的屏幕,而不考虑开关的置位)。在这前面必须有一个 HPLOT 语句。

(5)与游戏控制器及扬声器有关的命令

220　X＝PEEK(－ 16336)

触发(TOGGLE)扬声器一次,扬声器发出一个"卡嗒"声。

225　X＝PEEK(－ 16352)

触发盒式录音带输入一次,在盒式录音机上产生一个"卡嗒"声。

230　X =PEEK(−16287)

读 #0 游戏控制器上的按钮开关，如 X >127，那么按钮已被按了。

240　X =PEEK(16286)

同上面的一样，只是读 #1 游戏控制器的按钮开关。

250　X =PEEK(−16285)

读 #2 游戏控制器上的按钮。

(6)与出错有关的命令

340　X =PEEK(218)+ PEEK(219) * 256

如果一条 ONERR GOTO 语句已执行过，这条语句置 X 等于产生错误的那条语句的行号。

350　IF PEEK(216) >127 THEN GOTO 2000

如果内存 222 单元(ERRFLG)的第 7 位已经置为真，那么，已经遇到一第 ONERR GOTO 语句。

360　POKE 216,0

清除 ERRFLG，以便发出标准出错信息。

370　Y =PEEK(222)

置变量 Y 为描述出错类型的编码，这个错误已使得一条 ONERR GOTO 转移出现。错误类型描述见 ONERR GOTO 语句。

附录8　出错信息

在一个错误产生后，BASIC 返回到命令一级上，这由提示字符及一个闪烁的光标指出。变量和程序正文仍然是完整的，但是程序不能继续执行，并且所有的 GOSUB 和 FOR 循环计数器都置成 0，要在一个运行程序中避免这种中断，则可以把 ONERR GOTO 语句与错误处理例行程序结合起来使用。

当一个错误产生在一个立即执行语句中时，行号不被打印出来。出错信息的格式如下:

立即执行方式的语句　　?XX ERROR

间接执行方式的语句　　?XX ERROR IN YY

在上面的两个例子中，"XX"表示错误名，"YY"是产生错误的语句行号。一个间接执行语句中的错误，只有该语句被执行时才被发觉。

以下是可能产生的出错码以及它们的含义:

CAN'T CONTINUE

试图继续执行一个不存在的程序，或在产生出错后，试图继续执行程序，或者从程序中删除或增加一行之后，试图继续执行程序。

DIVISION BY ZERO

除数为零出错。

ILLEGAL DIRECT

你不能将 INPUT, DEF FN, GET, DATA 语句作为立即执行命令使用。

ILLEGAL QUANTITY

传送给数学函数或串函数的参数超出范围，产生 ILLEGAL QUANTITY 错误的情况可能是:

(a)负数为数组下标变量(例如: LET A(−1)=0)。

(b)用负数或零作为 LOG 的参数。

(c)用负数作 SQR 的参数。

(d)在 A ∧ B 中，A 为负数，且 B 不是一个整数。

(e)在 MID $, LEFT $, RIGHT $, WAIT, PEEK, POKE, TAB, SPC, ON...GOTO 或作图的任一函数中使用了不恰当的参数。

NEXT WITHOUT FOR

在一个 NEXT 语句中的变量名与当前的 FOR 语句中的变量名不一致，或者无变量名的 NEXT 语句不与任何 FOR 语句相对应。

OUT OF DATA

执行一个 READ 语句时，程序中的所有 DATA 语句早已被读完，程序试图要读更多的数据，但程序中所包含的数据不够。

OUT OF MEMORY

以下的任何一种情况都会产生这种信息：程序太长；变量太多；FOR 循环嵌套多于 10 层的深度；GOSUB 的嵌套深度超过 24 层；表达式太复杂；圆括号嵌套深度多于 36 层；企图置的 LOMEM：值太高；企图置的 LOMEM：值低于当前值；企图置的 HIMEM：值太低。

FORMULA TOO COMPLEX

执行到多于两条 IF "XX" THEN 形式的语句。

OVERFLOW

一个计算的结果太长，以致于不能用 BASIC 的数据格式表示，如果出现下溢出，给出一个零作为结果，并且继续执行，没有任何错误信息打印出来。

REDIM'D ARRAY

对一个数组两次使用 DIM 语句时，产生此错误。这个错误经常发生在这种情况下：如果直接使用象 A(I) =3 这样的语句，则 BASIC 自动定义其标量长度为 10，但在程序中又跟有如 DIM A(100) 的语句.如果你想要找出一个数组定义语句所在的程序行，这个出错信息是很有用的。这只要在第一行上插入该数组的定义语句 DIM，运行这个程序，则 BASIC 将告诉你原来的 DIM 语句是在什么位置上。

RETURN WITHOUT GOSUB

遇到了一条 RETURN 语句，但是没有执行相应的 GOSUB 语句。

STRING TOO LONG

试图通过使用连接运算产生一个多于 255 个字符的字符串。

BAD SUBSCRIPT

试图访问一个超出数组大小范围的数组元素，如果在一个数组的

访问中使用了错误维数，这种出错信息亦产生。例如：当 A 已经用 DIM A(2,2)定义了，但出现了 LET A(1,1,1) =Z 的语句。

SYNTAX ERROR

在一个表达式中，丢掉了圆括号，在一行中有不合法的字符，不正确的标点符号等等，均会产生此信息。

TYPE MISMATCH

一个赋值语句的左边是一个数值变量，而右边是一个串，或者左边是一个串变量，而右边是一个数值型数据，另一种情况是一个函数的参数类型是一个串，但给出的却是一个数值型数据，或者反过来。

UNDEF'D STATEMENT

试图使用 GOTO、GOSUB 使 THEN 转到一个行号不存在的语句上去。

UNDEF'D FUNCTION

调用了还未定义的用户定义函数。

附录9　字母、字符显示码表（基本字符组）

	反向显示		闪烁显示		正向显示		正向显示	
	大写字母	特殊字符	大写字母	特殊字符	大写字母	特殊字符	大写字母	小写字母
	$ 00–$ 1F	$ 20–$ 3F	$ 40–$ 5F	$ 60–$ 7F	$ 80–$ 9F	$ A0–$ BF	$ C0–$ DF	$ E0–$ FF
00	@		@		@		@	`
01	A	!	A	!	A	!	A	a
02	B	"	B	"	B	"	B	b
03	C	#	C	#	C	#	C	c
04	D	$	D	$	D	$	D	d
05	E	%	E	%	E	%	E	e
06	F	&	F	&	F	&	F	f
07	G	'	G	'	G	'	G	g
08	H	(	H	(	H	(	H	h
09	I	)	I	)	I	)	I	i
0A	J	*	J	*	J	*	J	j
0B	K	+	K	+	K	+	K	k
0C	L	,	L	,	L	,	L	l
0D	M	—	M	—	M	—	M	m
0E	N	.	N	.	N	.	N	n
0F	O	/	O	/	O	/	O	o

	反向显示		闪烁显示		正向显示		正向显示	
	大写字母	特殊字符	大写字母	特殊字符	大写字母	特殊字符	大写字母	小写字母
	$ 00–$ 1F	$ 20–$ 3F	$ 40–$ 5F	$ 60–$ 7F	$ 80–$ 9F	$ A0–$ BF	$ C0–$ DF	$ E0–$ FF
10	P		P				P	p
11	Q	1	Q	1	Q	1	Q	q
12	R	2	R	2	R	2	R	r
13	S	3	S	3	S	3	S	s
14	T	4	T	4	T	4	T	t
15	U	5	U	5	U	5	U	u
16	V	6	V	6	V	6	V	v
17	W	7	W	7	W	7	W	w
18	X	8	X	8	X	8	X	x
19	Y	9	Y	9	Y	9	Y	y
1A	Z	:	Z	:	Z	:	Z	z
1B	〔	;	〔	;	〔	;	〔	{
1C	＼	<	＼	<	＼	<	＼	|
1D	〕	=	〕	=	〕	=	〕	}
1E	∧	>	∧	>	∧	>	∧	~
1F	—	?	—	?	—	—	—	■

附录10　按入口地址排列的机内子程序

入口地址	功　　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的BASIC语句
$ F800	在低分辨率图形的第一页内画一个点。	将行数存入累加器A，列数存入变址寄存器Y。	没有	PLOT
$ F819	绘出一条低分辨率的水平线。	将行数存入累加器A中，左边的列数存入变址寄存器Y内，右边的列数存入存储器的44号单元内	A，Y	HLIN
$ F828	绘出一条低分辨率的垂直线。	列数放在变址寄存器Y内，上面的行数放在累加器A中，下边的行数放在存储器的45号单元内。	没有	VLIN
$ F832	将所有48行低分辨率图形的显示屏幕清除为黑色如果是课文模式，则用字符“@”填满整个屏幕	没有	A，Y	CALL－1998
$ F836	清除低分辨率图形的各行，但不变动课文显示窗口。	没有	A，Y	GR(见$ FB40)
$ F85F	将现有的低分辨率图形的颜色增加三种	没有	A	CALL－1953
$ F864	设置低分辨率图形的颜色。	代表颜色的编号放在累加器A中	A	COLOR

入口地址	功　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的BASIC语句
$ F871	将低分辨率图形中某一点的颜色读出。	行数放入累加器A中,列数放在变址寄存器Y中。	累加器A中存有代表颜色的编号	SCRN
$ F940	按YYXX的格式将变址寄存器Y及X的内容显示在屏幕上或输出到被选定的输出设备上	没有	没有	CALL-1728
$ F941	按AAXX的格式、将累加器A及变址寄存器X的内容送到显示屏幕上去显示,或输出到被选定的输出设备内	没有	没有	CALL-1727
$ F944	打印变址寄存器X的内容	没有	没有	CALL-1724
$ F948	输出三个空格到被选定的输出的设备内(根据CSW的内容来决定)。	没有	X、A	CALL-1720
$ F94A	输出1～256个空格到指定的输出设备内。	空格的数目存于累加器A中(存入0表示输出256个空格)	没有	SPC( ) CALL-1718
$ FB1E	读取摇杆0,1,2,或3的状态。	摇杆的编号存于变址寄存器X内。	变址寄存器Y内存有0-FF的数据,累加器A中的数据被清除。	PDL( )
$ FB2F	设置Apple-II的课文显示屏幕为24行,每行40个字符。	没有	A	TEXT

入口地址	功　　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的BASIC语句
$ FB40	设置低分辨率图形，并将显示屏幕清除，同时设置四行课文显示窗口	没有	A, Y	GR
$ FBD9	把字符 BELL 的（ASCII 码 7）送到被选定的输出设备内。	没有	A, Y	CALL-1063
$ FBE4	使 Apple-II 的扬声器发出 1/10 秒的"嘟"声	没有	A, Y	CALL- 1052
$ FC10	送一个空格返回字符到显示屏幕上，以修正光标所在位置。	没有	A	CALL- 1008
$ FC1A	将光标向上移动一行，如果光标已经在显示屏幕的顶部，则不做任何移动	没有	A	CALL- 998
$ FC42	将课文显示窗口中，从光标所在位置开始到显示屏幕的右下角为止的全部内容清除。	没有	A, Y	CALL- 958
$ FC46	将课文显示窗口中，从寄存器所指定的坐标开始，到显示屏幕右下角为止的全部内容清除。	列数存于变址寄存器 Y 内，行数存于累加器 A 内。	A, Y	CALL- 954
$ FC62	输出一个回车符号，并跳到显示屏幕的下一行。	没有		CALL- 926

入口地址	功　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的BASIC语句
$ FC66	将光标向下移动一行，但并不改变它的水平位置，如果光标在显示屏幕的最底部，则将显示的课文往上卷一行	没有	A，Y	CALL- 922
$ FC70	将课文显示窗口向上卷一行。	没有	A，Y	CALL- 912
$ FC9C	把从光标位置开始，到这一行的末尾所显示的课文全部清除，光标的位置保持不变	没有	A，Y	CALL- 868
$ FCA8	执行一个延时循环，它的延迟时间是0.5(5X + 27X+ 26)微秒	延迟的数值存放在累加器 A 内。	A	CALL-856
$ FD0C	当光标闪动时，等待键盘的输入，并把78 和 79 存储单元内的值作为随机数发生器的初始值。	没有	字符返回到累加器 A 中。受影响的寄存器：X，Y	CALL-756
$ FD35	除了还可以接受编辑码外，功能与$ FD0C 相同	没有	字符返回到累加器 A 中，受影响的寄存器：X，Y	CALL-715
$ FD67	输出一个回车符号到显示屏幕上，允许在同一行内最多具有256 个字符。	提示符存放在存储器的 51 号单元内。	Y，A。变址寄存器内存有输入数据的长度，数据输入从存储单元$ 2000 开始。	INPUT

入口地址	功　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的BASIC语句
$ FDDA	用二位十六进制数将累加器 A 中的值打印出来。	数据存放在累加器 A 中。	A	CALL-550
$ FDED	输出一个字符到现在被选定的输出设备内	字符存放在累加器 A 中。	没有	PRINT
$ FDF0	输出一个字符到Apple-II 课文显示窗口。	字符存放在累加器 A 中。	没有	PRINT
$ FE80	设置反极性的视频模式，即白底黑字的课文显示模式。	没有	Y	INVERSE CALL-384
$ FE84	设置正常的视频模式，即黑底白字的课文显示模式。	没有	Y	NORMAL CALL-380
$ FEB0	返回 BASIC 语言系统，抹掉存储器中的程序和变量。	没有	A,X,Y	CALL-336
$ FF3F	将寄存器中的内容重新存入寄存器(只有在预先执行了由 $ FF4A 开始的机内子程序后方为有效)。	没有	将各寄存器的内容存入下列单元内：累加器 A: 69($ 45)，寄存器 S: 72($ 48)，变址寄存器 X: 70($ 46)，堆栈指示器 SP: 73($ 49)，变址寄存器 Y: 71($ 47)	CALL-193

入口地址	功　　能	调用前必须使用的寄存器	受影响的寄存器	相同功能的 BASIC 语句
$ FC58	将整个课文屏幕的显示清除，并把光标移到显示屏幕的左上角。	没有	A，Y	HOME CALL－936
$ FF2D	打印 ERR 信息，并使主机板上的扬声器发出“嘟”的一声。	没有	A	CALL－211
$ FF3A	主机板上的扬声器发出“嘟”的一声。	没有	A	CALL－198
$ FF4A	将存储器的内容存入到第 0 页的下述各单元内，累加器 A：69（$ 45）寄存器 S：72（$ 48）变址寄存器 X：70（$ 46）堆栈指示器 SP：73（$ 49）变址寄aw存器 Y：71（$ 47）	没有	没有	CALL－182
$ FF69	监控系统的入口地址	没有	没有	CALL－151

具有上标 A 的 BASIC 命令，只能在 Applesoft 语言系统中使用。

附录11　CEC-LOGO命令参考表

命令功能	简释	缩写
ALLOF	全部参数为真与否	
ANYOF	任一个参数为真与否	
ASCII	求字符的 ASCII 码	
ATAN	求参数的反正切	
BACK	图标后移	BK
BACKGROUND	屏幕之背景色	BG
BUTFIRST	取出除第一个元素外的表(词)	BF
BUTLAST	取出除最后一个元素外的表(词)	BL
CATALOG	列出磁盘目录	
CHAR	由 ASCII 码输出其相应字符	
CLEARINPUT	清洗输入字符缓冲区	
CLEARSCREEN	清图形屏幕	CS
CLEARTEXT	清文本屏幕且光标复位	
CONTINUE	暂停后恢复运行	
COS	求参数的余弦值	
CURSOR	将光标移到相应的列、行	
DOS	执行 DOS 命令行	
DRAW	清屏、进入绘图模式	
EDIT	进入编辑过程	ED
ELSE	IF...THEN...ELSE 的部分	
ERASEFILE	删除磁盘上的过程文件	
ERASEPICT	删除磁盘上的图形文件	
END	过程定义的结束符	
ERASE	删除工作区中某过程	ER
ERNAME	删去变量(可以是表、词或数)名	
FIRST	把第一个参数放到第二个表前面	
FORWARD	图标向前移动	FD
PUT	把第一个参数放到第二个表前面	
FULLSCREEN	设置全屏幕绘图模式	

GO	转向参数所指标号	
GOODBYE	清工作区，并重新启动 LOGO	
HEADING	当前图标的朝向	
HIDETURTLE	隐去图标	HT
HOME	使图标移到起始位置	
IF	条件语句的开始	
IFFALSE	如果测试结果为假	IFF
IFTRUE	如果测试结果为真	IFT
LAST	取出参数(表、词)的最后一个元素	
LEFT	使图标向左转	LT
LIST	将各参数(词、表或数)合起一个表	
LIST?	判断参数是否为一个表	
LPUT	把第一个参数放到第二个表的后面	
MAKE	给变量赋值	
NODRAW	退出绘图模式	ND
NOT	输出参数的否定值	
NOTRACE	停止跟踪	
NOWRAP	停止卷动状态	
NUMBER?	判断参数是否为数	
OUTDE	定义输出设备	
OUTPUT	传送信息并退出所在过程	OP
PADDLE	检测某游戏操作杆的角度	
PADDLEBUTTON	检测游戏按钮是否按下	
PAUSE	暂停过程执行	
PENCOLOR	设置笔色	PD
PENDOWN	落笔	PD
PENUP	提笔	PU
PRINT	打印并换行	PR
PRINT 1	打印但不换行	
PRINTOUT	打印过程清单	PO
QUOTIENT	输出两参数的整数商	
RANDOM	产生随机数	

RANDOMIZE	使随机数更随机化	
RC?	检测是否有字符输入	
READ	从磁盘中读出文件装入工作区	
READCHARACTER	读入一个字符	RC
READPICT	从磁盘上读出图形装入图形区	
REMAINDER	求二数相除的余数	
REPEAT	重复执行某一命令行	
REQUEST	请求输入一行数据	RQ
RIGHT	使图标向右转	RT
ROUND	四舍五入法取整	
SAVE	把工作区全部过程存入磁盘	
SAVE PICT	把图形存入磁盘	
SENTENCE	句子合成	SETH
SETHEADING	设置图标朝向	
SETX	水平移图标到指定横坐标	
SETXY	移动图标到指定位置	
SETY	垂直移图标到指定纵坐标	
SHOWTURTLE	显示图标形状	ST
SIN	求参数的正弦	
SPLITSCREEN	设置图形 / 文本混合模式	
SQRT	求参数的平方根	
STOP	停止执行当前过程	
TEST	检测并保留其结果,供 IFF 与 IFL 使用	
TO	过程的首部	
TOPLEVEL	退回最外层控制状态	
TOWARDS	将图标方向角置为指向定点	
TRACE	开始跟踪	
TURTLESTATE	输出图标的当前状态	TS
WORD	把词或字符组成新词	
WORD?	检测参数是否为词	
X COR	输出图标当前位置 X 坐标	
Y COR	输出图标当前位置 Y 坐标	

- ASPECT　决定图形屏幕纵横向比例因子
- CALL　调用机器语言子程序
- DEPOSIT　送一数值到指定内存单元
- EXAMINE　输出某一内存单元的值
- GCOLL　强制废料收集
- NOPES　检查有多少自由结点

ISBN 7—5053—0237—X/TP29

CEC—Ⅰ型中华学习机用户使用手册